'n Masjienleerbenadering tot woordafbreking in Afrikaans

deur

Machteld Fick

voorgelê luidens die vereistes vir die graad

Doctor Philosophiæ

in die vak

Operasionele Navorsing

aan die

Universiteit van Suid-Afrika

Studieleier: Professor CJ Swanepoel

Junie 2013

Opsomming

Die doel van hierdie studie was om te bepaal tot watter mate 'n suiwer patroongebaseerde benadering tot woordafbreking bevredigende resultate lewer. Die masjienleertegnieke *kunsmatige neurale netwerke*, *beslissingsbome* en die TEX-algoritme is ondersoek aangesien dit met letterpatrone uit woordelyste afgerig kan word om lettergreep- en saamgesteldewoordverdeling te doen.

'n Leksikon van Afrikaanse woorde is uit 'n korpus van elektroniese teks genereer. Om lyste vir lettergreep- en saamgesteldewoordverdeling te kry, is woorde in die leksikon in lettergrepe verdeel en saamgestelde woorde is in hul samestellende dele verdeel. Uit elkeen van hierdie lyste van ±183 000 woorde is ±10 000 woorde as toetsdata gereserveer terwyl die res as afrigtingsdata gebruik is.

'n Rekursiewe algoritme is vir saamgesteldewoordverdeling ontwikkel. In hierdie algoritme word *alle* ooreenstemmende woorde uit 'n verwysingslys (die leksikon) onttrek deur stringpassing van die begin en einde van woorde af. Verdelingspunte word dan op grond van woordlengte uit die samestelling van begin- en eindwoorde bepaal. Die algoritme is uitgebrei deur die tekortkominge van hierdie basiese prosedure aan te spreek.

Neurale netwerke en beslissingsbome is afgerig en variasies van beide tegnieke is ondersoek om die optimale modelle te kry. Patrone vir die TEX-algoritme is met die OPatGen-program gegenereer. Tydens toetsing het die TEX-algoritme die beste op beide lettergreep- en saamgesteldewoordverdeling presteer met 99,56% en 99,12% akkuraatheid, respektiewelik. Dit kan dus vir woordafbreking gebruik word met min risiko vir afbrekingsfoute in gedrukte teks. Die neurale netwerk met 98,82% en 98,42% akkuraatheid op lettergreep- en saamgesteldewoordverdeling, respektiewelik, is ook bruikbaar vir lettergreepverdeling, maar dis meer riskant. Ons het bevind dat beslissingsbome te riskant is om vir lettergreepverdeling en veral vir woordverdeling te gebruik, met 97,91% en 90,71% akkuraatheid, respektiewelik.

'n Gekombineerde algoritme is ontwerp waarin saamgesteldewoordverdeling eers met die TEX-algoritme gedoen word, waarna die resultate van lettergreepverdeling deur beide die TEX-algoritme en die neurale netwerk gekombineer word. Die algoritme het 1,3% minder foute as die TEX-algoritme gemaak. 'n Toets op gepubliseerde Afrikaanse teks het getoon dat die risiko vir woordafbrekingsfoute in teks met gemiddeld tien woorde per reël ±0,02% is.

Sleutelwoorde: woordafbreking, lettergreepverdeling, saamgesteldewoordverdeling, stringpassing, woordvlakakkuraatheid, verdelingsgeleentheidsvlakakkuraatheid, masjienleertegnieke, neurale netwerke, beslissingsbome, algoritme

Abstract

The aim of this study was to determine the level of success achievable with a purely pattern based approach to hyphenation in Afrikaans. The machine learning techniques *artificial neural networks*, *decision trees* and the TEX algorithm were investigated since they can be trained with patterns of letters from word lists for syllabification and decompounding.

A lexicon of Afrikaans words was extracted from a corpus of electronic text. To obtain lists for syllabification and decompounding, words in the lexicon were respectively syllabified and compound words were decomposed. From each list of ±183 000 words, ±10 000 words were reserved as testing data and the rest was used as training data.

A recursive algorithm for decompounding was developed. In this algorithm *all* words corresponding with a reference list (the lexicon) are extracted by string fitting from beginning and end of words. Splitting points are then determined based on the length of reassembled words. The algorithm was expanded by addressing shortcomings of this basic procedure.

Artificial neural networks and decision trees were trained and variations of both were examined to find optimal syllabification and decompounding models. Patterns for the TEX algorithm were generated by using the program OPatGen. Testing showed that the TEX algorithm performed best on both syllabification and decompounding tasks with 99,56% and 99,12% accuracy, respectively. It can therefore be used for hyphenation in Afrikaans with little risk of hyphenation errors in printed text. The performance of the artificial neural network was lower, but still acceptable, with 98,82% and 98,42% accuracy for syllabification and decompounding, respectively. The decision tree with accuracy of 97,91% on syllabification and 90,71% on decompounding was found to be too risky to use for either of the tasks

A combined algorithm was developed where words are first decompounded by using the TEX algorithm before syllabifying them with both the TEX algoritm and the neural network and combining the results. This algoritm reduced the number of errors made by the TEX algorithm by 1,3% but missed more hyphens. Testing the algorithm on Afrikaans publications showed the risk for hyphenation errors to be ±0,02% for text assumed to have an average of ten words per line.

Key words: hyphenation, syllabification, decompounding, string fitting, word level accuracy, splitting opportunity level accuracy, machine learning, neural networks, decision trees, algoritm

Bedankings

Dit sou nie vir my moontlik gewees het om hierdie proefskrif te voltooi sonder die hulp en bystand van wonderlike mense en die liefde en genade van my hemelse Vader nie.

Ek wil veral my man, Lourens, bedank vir sy aanmoediging, geduld en persoonlike ondersteuning in moeilike tye. Ook aan my kinders Suné, Coenraad en Nettie, my ma Sara en my sussie Ronéll wat my deurgaans aangemoedig en bygestaan het, baie dankie.

Hierdie proefskrif sou nie moontlik gewees het sonder die hulp en ondersteuning van my studieleier, Prof. Chris Swanepoel, nie. Sy fenomenale kennis en onuitputlike bron van idees het my pad tot hier 'n avontuur gemaak. Hy was altyd bereid om te help en daarvoor sal ek hom ewig dankbaar wees.

Laastens wil ek die Universiteit van Suid-Afrika bedank vir sabbatsverlof waartydens ek groot vordering met my studies kon maak. Ook aan my kollegas by die Departement Besluitkunde, dankie vir die belangstelling en vriendelikheid.

Inhoudsopgawe

Hoofstuk 1

Inleiding

In die rekenaargedrewe wêreld van vandag is dit belangrik dat woordverwerkingsprosesse soos woordafbreking outomaties gedoen kan word. Vir 'n taal soos Engels is daar reeds baie navorsing gedoen om tegnieke te ontwikkel om hierdie taak te verrig en die probleem is tot 'n groot mate reeds opgelos.

Vir tale soos Afrikaans, Nederlands en Duits waar saamgestelde woorde aanmekaar geskryf word, bestaan daar egter ruimte om alternatiewe metodes te ondersoek om die gehalte van woordafbrekers te verbeter. In sulke tale kan nuwe woorde na willekeur geskep word deur bloot twee (of meer) woorde aanmekaar te skryf en dis onmoontlik om 'n volledige leksikon saam te stel. Met die skep van sulke nuwe woorde word daar geduring nuwe letterpatrone geskep rondom die punte waar die woorde weer tydens woordafbreking verdeel moet word. Dit is dus nie moontlik om vaste woordafbrekingsreëls daar te stel nie en ander maniere moet gevind word om die probleem aan te spreek.

Vir iemand wat vaardig is in 'n taal soos Afrikaans, waar woordafbreking volgens lettergreep-verdeling gedoen word, is dit nie moeilik om 'n woord in lettergrepe te verdeel deur dit bloot lettergreep-vir-lettergreep uit te spreek nie. Op hierdie manier word natuurlike breekpunte in woorde geïdentifiseer wat ook met die afbrekingsposisies in woorde ooreenstem. 'n Rekenaar sien woorde egter bloot as stringe karakters, sonder enige aanduiding van waar die breekpunte behoort te wees.

Vroeër, voordat rekenaars in die drukkersbedryf gebruik is, is lettersetwerk per hand gedoen. Die operateur sou gewaarsku word wanneer die einde van 'n reël nader kom en hy kon dan besluit of dit nodig is om die betrokke woord af te breek en in watter posisie. Met die koms van rekenaars en die behoefte om woordafbreking outomatiese te doen, het hierdie voordeel egter verdwyn en moet rekenaartegnieke gebruik word vir hierdie taak. By samestellende tale is dit veral belangrik dat 'n "intelligente" woordafbreekalgoritme daargestel word om hierdie besluite te neem.

Die doel van hierdie projek is om te bepaal tot watter mate dit moontlik is om woordafbreking in Afrikaans op 'n meganiese manier te doen, deur bloot die letters in woorde en die patrone wat dit vorm, te gebruik. Woorde word dus geïsoleerd beskou met geen verwysing na kon-

teks of sintaksis[1] nie. Ons wil vasstel wat die beperkings is van masjienleertegnieke wat slegs letterpatrone in woorde gebruik om woordafbreking te doen.

In hierdie studie volg ons 'n pragmatiese benadering om te bepaal watter masjienleertegniek die beste resultate vir woordafbreking in Afrikaans lewer, en beoog nie om aan te toon dat een tegniek statisties beduidend beter as 'n ander presteer nie. Ons gebruik deurgaans 'n datastel van ±183 000 woorde waaruit ±10 000 woorde vir die toets van ontwikkelde modelle gereserveer word. Alle toetsing word met hierdie toetsdatastel gedoen.

Verder is dit belangrik om daarop te let dat die outeur nie 'n taalkundige is nie, maar 'n operasionele navorser met 'n belangstelling in die Afrikaanse taal.

In die volgende afdelings bespreek ons inleidend belangrike aspekte van die studie, naamlik algemene agtergrond oor Afrikaans en die Germaanse tale, woordafbreking met verwysing na lettergreep- en saamgesteldewoordverdeling en masienleertegnieke.

1.1 Afrikaans

Afrikaans is 'n Germaanse taal wat in Suid-Afrika uit die 17de eeuse Nederlands ontwikkel het. Dit word saam met Nederlands as 'n Nederfrankiese taal geklassifiseer. Nagenoeg 95% van die Afrikaanse woordeskat is van Nederlandse oorsprong [Wik13b, Stu13]. Sedert die koms van Jan van Riebeeck in 1652 was Suid-Afrika 'n multikulturele land, met 'n verskeidenheid tale wat gepraat is. Die Nederlanders het hulle hier gevestig; die Nederlandse Verenigde Oos-Indiese Kompanje het Oosterlinge uit Indië en Indonesië as slawe na die Kaap gebring [Mie13]; die Franse Hugenote het hierheen gevlug; ensovoorts. Die tale van hierdie volke, asook inheemse stamme soos die Khoisan en swart volke soos die Zoeloes en Xhozas het almal 'n invloed op die ontwikkeling van Afrikaans gehad.

Voorbeelde van woorde wat nie uit Nederlands kom nie, maar algemeen in Afrikaans voorkom, is "baie" wat aan Maleis ontleen is, "gogga" uit Khoisan, "bredie" en "kombers" uit Portugees en "moetie", "haikôna" en "kaia" uit inheemse Afrikatale [Wik13b].

Sedert 1875 toe die *Genootskap van Regte Afrikaners* gestig is, het die ontwikkeling en standardisering van Afrikaans momentum gekry.

Reeds in 1905 is die eerste verskille tussen Nederlandse en Afrikaanse spelling uitgespel in die *Vereenvoudigde Nederlandse Spelling in Suid-Afrika*. Later, in 1917 verskyn die eerste *Afrikaanse Woordelys en Spelreëls* (AWS) van die *Suid-Afrikaanse Akademie vir Wetenskap, Lettere en Kuns* waarin die eerste grondbeginsels vir Afrikaans verskyn het. Hierdie eerste AWS het die Grondbeginsels van Afrikaanse spelling uiteengesit, met beginsels soos dat elke klank deur 'n aparte letter voorgestel moet word, geen onnodige letters gebruik behoort te word nie en dat die gebruiklikste uitspraak as norm aanvaar word [Wik13c]. Daar het reeds tien uitgawes van die AWS verskyn, met die nuutste een in 2009.

[1]Sintaksis – die struktuur van 'n sin, insluitend woordsoorte en woordvolgorde.

Spelling in Afrikaans is baie dieselfde as in Nederlands, behalwe dat sekere aanpassings gemaak is, hoofsaaklik as gevolg van uitspraak. Voorbeelde hiervan is konsonante wat weggelaat is, soos die "g" in *spiegel* of *vogel*; die "z" wat in Afrikaans as 'n "s" uitgespreek word, soos in *zeg* en *zorg*; klinkerkombinasies wat deur 'n enkele klinker vervang is, soos "ij" in *ontbijt*; ens. In Tabel 1.1 word voorbeelde van sulke ortografiese[2] veranderings getoon.

Konsonante								**Vokale**			
verander				**weggelaat**				**verander**			
Letters		**Voorbeelde**		**Letters**		**Voorbeelde**		**Letters**		**Voorbeelde**	
Ndl	Afr	Ndl	Afr	Ndl	Afr	Ndl	Afr	Ndl	Afr	Ndl	Afr
c	k	cultuur	kultuur	g	-	spiegel	spieël	ij	y	ontbijt	ontbyt
z	s	zorg	sorg			vogel	voël	ij	i	persoonlijk	persoonlik
v	w	liever	liewer			zeg	sê	ion	ie	station	stasie
ch	g	dochter	dogter	n	-	leven	lewe	ouw	ou	vrouw	vrou
cht	g	slecht	sleg			haven	hawe	auw	ou	dauw	dou
sch	sk	school	skool	g en n	-	morgen	môre	je	ie	koekje	koekie
th	t	thuis	tuis			bruggen	brûe				
tie	sie	politie	polisie								
cie	sie	provincie	provinsie								

Tabel 1.1: Verskille tussen Nederlandse en Afrikaans spelling

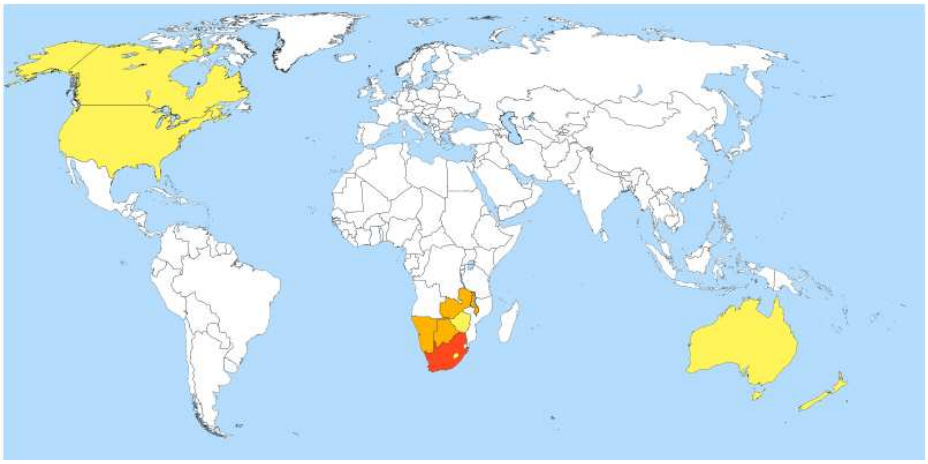
Afrikaans is een van die jongste erkende tale. In 1925 is dit grondwetlik aan Nederlands gelyk gestel en het dit dus amptelike status gekry [Wik13b, SAH13]. Naas Faroëes en Luxemburgs is dit die derde jongste Germaanse taal wat amptelike status geniet.

Afrikaans is een van die elf amptelike tale in Suid-Afrika met nagenoeg 6,59 miljoen moedertaalsprekers wat 13,5% van die bevolking uitmaak. In Namibië gebruik 11% van huishoudings Afrikaans as eerste taal, en in die groter Suider Afrika gebruik ongeveer 10,3 miljoen mense Afrikaans as tweede of derde taal.

Hoofsaaklik as gevolg van emigrasie kom daar Afrikaanssprekende mense ook in ander dele van die wêreld, soos die Verenigde Koninkryk, Nieu-Zeeland, Botswana, Australië en Kanada voor. In Figuur 1.1 word die wêreldverspreiding van Afrikaans getoon. In die geel gebiede op die kaart is Afrikaanssprekers 'n minderheid, in die oranje gebiede word dit gereeld gebruik, maar sonder amptelike status, en in die rooi gebied word Afrikaans algemeen gebruik en beskik dit oor amptelike status.

Aangesien Afrikaans 'n Germaanse taal is, verskaf ons vervolgens ook inligting daaroor.

[2]Ortografie – die gestandardiseerde manier waarop 'n taal geskryf word. Die woord ortografie word dikwels as sinoniem vir spelling gebruik, maar dit sluit ook taalelemente soos woordafbreking, die gebruik van hoofletters, beklemtoning en punktuasie in.

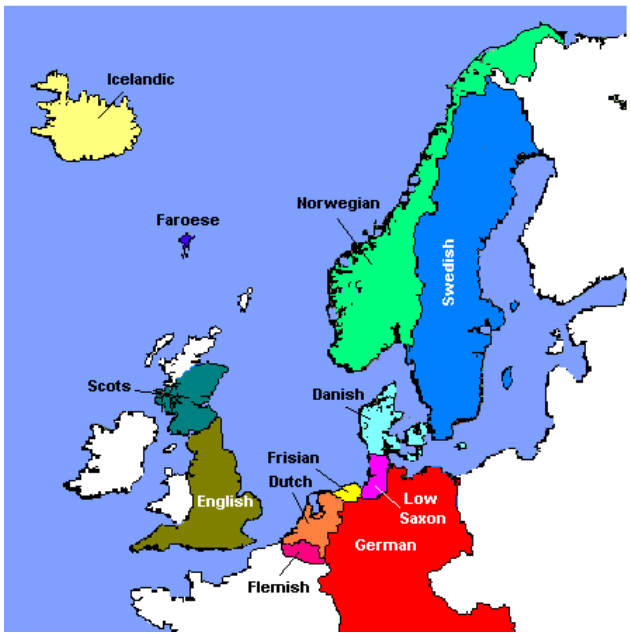


Figuur 1.1: Wêreldverspreiding van Afrikaans [Wik13b]

1.1.1 Germaanse tale

Die Germaanse tale is 'n vertakking van die Indo-Europese taalfamilie. Dit bestaan uit drie groepe, naamlik Wes-Germaans wat Engels, Duits en Nederlands insluit; Noord-Germaans wat Deens, Sweeds, Yslands, Noorweegs en Faroëes insluit en Oos-Germaans wat Goties en die tale van die Vandale, Boergondiërs en sommige ander stamme insluit. Laasgenoemde groep het uitgesterf.

Afrikaans is 'n Germaanse taal wat hoofsaaklik aan die suidpunt van Afrika gepraat word, wat buite die hoofgebied val waar die Germaanse tale voorkom. Figuur 1.2 toon hierdie hoofgebied.



Figuur 1.2: Verspreiding van Germaanse tale [Wik13e]

In Tabel 1.2 word die moderne standaard Germaanse tale en die benaderde aantal moedertaalsprekers van elk getoon [Buc13].

Wes-Germaans		**Noord-Germaans**	
Engels	375 000 000	Deens	5 000 000
Duits	98 000 000	Noorweegs	4 000 000
Nederlands	23 000 000	Frisies	400 000
Afrikaans	7 000 000	Jiddisj	400 000
		Yslands	260 000
		Faroëes	44 000

Tabel 1.2: Moedertaalsprekergetalle van moderne, standaard Germaanse tale

In hierdie studie konsentreer ons op woordafbreking in Afrikaans en bespreek dit met verwysing na lettergreep- en saamgesteldewoordverdeling in die volgende afdelings.

1.2 Woordafbreking

Reeds met die druk van die eerste boeke het drukkers besef dat dit nodig is om 'n woord wat nie aan die einde van reël inpas nie te verdeel, 'n koppelteken in te voeg en die res dan op die volgende reël te druk, ten einde egaligheid in die teks te verkry. Johann Gutenberg het byvoorbeeld reeds in die 1450s gereeld van afbreking gebruik gemaak, soos in die uittreksel uit sy "42-reël-Bybel" in Figuur 1.3 gesien kan word.

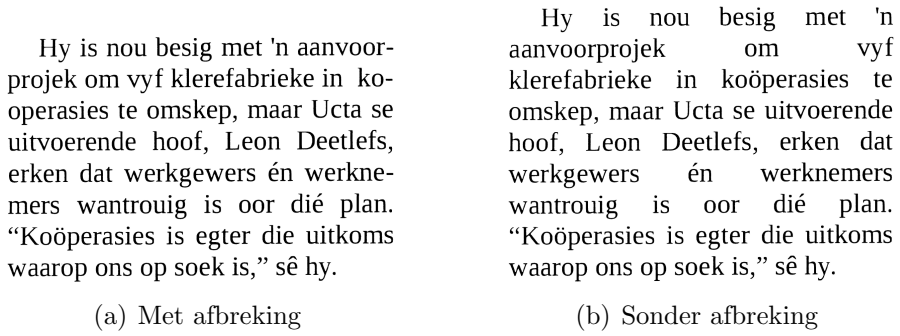
Figuur 1.3: Uittreksel uit Gutenberg se 42-reël Bybel [Ima13]

Tydens moderne woordverwerking gebeur dit dikwels dat daar te min plek aan die einde van 'n reël oor is vir 'n volledige woord om in te pas. Ten spyte daarvan dat beperkte addisionele witspasie soms toegelaat word, is dit dikwels nodig om so 'n woord dan óf af te breek, óf na die volgende reël oor te dra.

Die volgende is twee voorbeelde van reëls rakende woordafbreking wat in die Afrikaanse Woordelys en Spelreëls (AWS09) [Die09] voorkom:

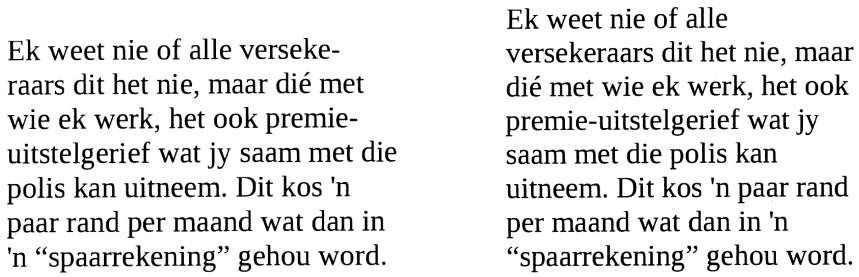
(a) *'n Woord wat nie aan die einde van 'n reël inpas nie, moet by voorkeur volledig na die volgende reël geskuif word.*

Indien dit egter 'n lang woord is, veroorsaak dit by dubbelgeskouerde teks dat groot spasies tussen woorde gevorm word wat drukwerk 'n onegalige voorkoms gee, soos getoon in Figuur 1.4(b). By linksgeskouerde teks het dit weer erg variërende regterkante tot gevolg soos getoon in Figuur 1.5(b).



(a) Met afbreking (b) Sonder afbreking

Figuur 1.4: Dubbelgeskouerde teks



(a) Met afbreking (b) Sonder afbreking

Figuur 1.5: Linksgeskouerde teks

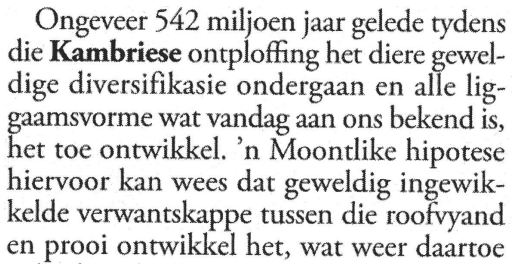
Woordafbreking is dus nodig, veral in koerante en tydskrifte waar teks in smal kolomme gedruk word.

(b) *Woordafbreking moet op grond van betekenisvolle dele en lettergrepe gedoen word, terwyl herkenbaarheid en interpreteerbaarheid in aanmerking geneem word.*

Wanneer woordafbreking op grond van betekenisvolle dele gedoen word, word tussen voor- of agtervoegsels en die stamwoord afgebreek (*be-man, man-lik*) en saamgestelde woorde word tussen die samestellende woorde afgebreek (*skryf-werk, papier-dun*).

Wanneer woorde suiwer op grond van lettergreepverdeling afgebreek word, kan 'n woord soos *let-ter-greep-ver-de-ling* by enigeen van die verdelingspunte afgebreek word. Indien herkenbaarheid en interpreteerbaarheid egter in aanmerking geneem word, is vorme soos *let-tergreepverdeling* en *lettergreepverde-ling* nie aan te beveel nie.

Uit die voorbeeld in Figuur 1.6 is dit egter duidelik dat hierdie aanbeveling nie streng nagevolg word nie.


Ongeveer 542 miljoen jaar gelede tydens die **Kambriese** ontploffing het diere geweldige diversifikasie ondergaan en alle liggaamsvorme wat vandag aan ons bekend is, het toe ontwikkel. 'n Moontlike hipotese hiervoor kan wees dat geweldig ingewikkelde verwantskappe tussen die roofvyand en prooi ontwikkel het, wat weer daartoe

Figuur 1.6: Herkenbaarheid nie in ag geneem nie (Huisgenoot, 21 Maart 2013)

Verder word die volgende stelling in die AWS gemaak:

> *Hoewel afbreking in drukwerk meestal outomaties deur rekenaarprogramme beheer word, word leiding hier gegee oor die korrekte afbreekstelsel.* [Die09]

Hierdie stelling is egter ietwat voortydig, aangesien daar nog nie 'n rekenaartegniek bestaan wat betroubaar genoeg is om woordafbreking in Afrikaans te doen en dit as korrek aanvaar kan word nie.

Die woordafbreker wat by die *Pharos E-speller: Speltoetser en woordafbreker vir Afrikaans* [Pha13] ingesluit is en die een wat by *Afrikaanse Skryfgoed 4* [NP13] ingesluit is, word in Microsoft Office gebruik. By gebrek aan die programmatuur om dit voluit mee te toets, het ons woorde uit ons datastel daarmee getoets en bevind dat beide dikwels (dieselfde) foute maak, veral by die grense tussen die samestellende dele van saamgestelde woorde. Voorbeelde hiervan is *boe-killustrasies*, *prinsloos-traat*, *vinge-rete*, *sni-kwarm*, *xhosat-jie*. Verder mis dit ook dikwels verdelings in woorde, soos *han-delsper-seel*, *har-singskud-ding*, *winsoog-merk*. Alhoewel dit algemene woorde goed hanteer, kan dit ongelukkig nie vertrou word om woordafbreking *foutloos* te doen nie.

Om te verseker dat woordafbrekingsfoute nie in gedrukte teks voorkom nie, word woordafbreking gewoonlik per hand gekontroleer voordat boeke, koerante en tydskrifte gepubliseer word. Mense is feilbaar en foute kom steeds in drukwerk voor, soos die voorbeelde uit gepubliseerde boeke en tydskrifte in Figuur 1.7 illustreer.

Om tyd te spaar en om afbrekingsfoute te voorkom, word woordafbreking deesdae in tydskrifte vermy deur linksgeskouerde teks met variërende regterkante te gebruik (Figuur 1.8(a)). Om dieselfde rede word teks selfs in gesentreerde kolomme gedruk om woordafbreking te vermy (Figuur 1.8(b)).

'n Outomatiese woordafbreker spesifiseer normaalweg diskresionêre afbreekpunte wat na gelang van die teksuitleg gebruik word. Indien afbreking per hand reggestel word deur 'n "harde" koppelteken in te plaas, mag dit gebeur dat 'n so 'n koppelteken in die teks agterbly wanneer die uitleg verander. Figuur 1.9 toon 'n voorbeeld hiervan.

Na gelang van die font en fontgrootte wat vir 'n dokument gebruik word, is elke woord potensieel 'n kandidaat vir woordafbreking. Vir die beste resultate moet soveel moontlik geldige

sou bly kleef, en hy het later gesê dat dit hom gepas het dat mense nie dramatiese veranderinge van hom verwag het nie, terwyl hy reeds gewéet het watter rigting hy wou inslaan.

(a) *Elita en haar lewe met FW de Klerk* deur Martie Retief Meiring, 2008

In hierdie bydrae word gepoog om 'n algemene raamwerk te beskryf waarin berekenbaarheid vir 'n willekeurige topologiese ruimte X gedefinieer kan word. Die elemente van X word in dié benadering beskou as die primitiewe voorwerpe – ook vir die kodering van funksies. Ons begin deur nodige

(b) Die Suid-Afrikaanse Tydskrif vir Natuurwetenskap en Tegnologie, Desember 2008 (p. 288)

Links van hulle, oorkant die paadjie, sit 'n jong paartjie met 'n baba en 'n vrou langs hulle wat onbedaarlik

(c) Rooi Rose, September 2009

BABA-AART-APPELTJIES GEBRAAI IN EENDVET

(d) Sarie Kos, 2013

Figuur 1.7: Voorbeelde van afbrekingsfoute in publikasies

"My werk het vir my 'n vorm van terapie geword. Veral die mense van die hospies het my en my suster aangeraai om by ondersteuningsgroepe aan te sluit om ons te

(a) Rooi Rose, Mei 2013

In 2010 het hy 'n miljoen dollar geskenk aan fondse wat gestig is om **slagoffers van aardskuddings** te help, asook 'n miljoen dollar aan die **Wildlife Conservation Society**.

(b) Rooi Rose, Junie 2013

Figuur 1.8: Vermyding van woordafbreking

Lettres. En tog kan al die toekennings in die wêreld haar op hier-die laat-Vrydagmiddag in Boston, Kaapstad, nie skeel nie. Ná nog 'n stampvol week het sy net oë vir Adam en hy vir haar.

Figuur 1.9: Koppelteken per hand ingeplaas (Rooi Rose, Mei 2013)

afbrekingspunte gedek word, wat natuurlik met lettergreepverdelingsposisies ooreenstem.

Die hoofdoel van hierdie studie is om 'n metode te ontwikkel om outomaties woordafbrekingspunte vir Afrikaanse woorde te bepaal. Aangesien woordafbreking volgens lettergrepe gedoen word, is lettergreepverdeling 'n belangrike taak in hierdie proses. Verder is dit vir enige samestellende taal ook belangrik om die grense tussen die samestellende dele van saamgestelde woorde te bepaal, wat 'n belangrike aspek is van lettergreepverdeling en by implikasie ook van woordafbreking.

1.2.1 Lettergreepverdeling

Ten einde 'n onbekende woord uit te spreek, breek mens gewoonlik die woord in kleiner dele op en spreek dit so uit. Kinders leer ook lees deur gedeeltes van woorde te "klank" [BKC08]. Hierdie klein, uitspreekbare gedeeltes van woorde word *lettergrepe* genoem en dit stem dikwels met morfeme[3] ooreen.

Natuurlike-taalverwerkingstake soos spraakherkenning, teks-na-spraak-sintese, teks-na-foneem-omskakeling en ook woordafbreking tydens woordverwerking is grootliks op lettergreepverdeling gebaseer. Vir hierdie take is dit dus nodig om 'n tegniek te ontwikkel wat lettergreepverdeling outomaties kan doen [MAD09].

Baie navorsing is reeds oor outomatiese lettergreepverdelingstegnieke vir Duits [BKS00, KS03] en Nederlands [DvdB92] gedoen en natuurlik ook vir Engels [BKC08, Lia83] waarvoor die probleem tot 'n groot mate reeds opgelos is. Daar is egter ruimte vir die verbetering van sulke tegnieke vir Afrikaans.

Rekenaarmatige lettergreepverdeling

Daar is twee benaderings tot rekenaarmatige lettergreepverdeling, naamlik reëlgebaseerde en datagedrewe metodes [MAD09]. Alhoewel daar reëls vir lettergreepverdeling in Afrikaans bestaan, is dit nie moontlik om al sulke reëls in 'n rekenaarprogram vas te vang nie. 'n Reël soos *verdeel tussen twee identiese medeklinkers wat langs mekaar tussen klinkers staan* is wel programmeerbaar, maar dis nie moontlik om 'n vae, ondefinieerbare reël soos *breek tussen betekenisvolle dele* in programmeertaal te omskryf nie. Ten einde 'n metode te ontwikkel wat hopelik sulke reëls sal ondervang, konsentreer ons op datagedrewe masjienleertegnieke.

SiSiSi[4] is 'n lettergreepverdelingstegniek wat 'n tabel van moontlike woorddele, soos voorsetsels, stamme en agtervoegsels gebruik om woorde volgens woordverbuigings- en woordsamestellingsreëls in lettergrepe te verdeel [Bar02]. Slegs wanneer daar geen twyfel bestaan dat 'n lettergreepverdelingsposisie geldig is nie, word dit vir woordafbreking gebruik.

In teenstelling met sulke tegnieke ondersoek ons die moontlikheid om lettergreepverdeling te doen deur slegs die letters waaruit woorde bestaan en die patrone wat daardeur gevorm word, in aanmerking te neem. Woorde word dus in isolasie beskou, sonder enige verwysing na die wyer konteks, morfologie[2] of sintaksis. Ons doel is om op 'n pragmatiese wyse te bepaal tot watter mate outomatiese lettergreepverdeling deur middel van 'n patroongebaseerde proses gedoen kan word. Hiervoor word masjienleertegnieke aangewend wat vir patroonherkenning afgerig word.

[3] 'n Morfeem is die kleinste linguistiese eenheid van vorm-met-betekenis.

[4] Akroniem vir *Si*chere *Si*nnensprechende *Si*lbertrennung (betroubare, betekenisdraende woordafbreking).

[2] Morfologie – die struktuur van 'n woord, insluitend verbuigings- en vervoegingsvorme.

1.2.2 Saamgesteldewoordverdeling

In Afrikaans, soos in ander samestellende tale, kan bestaande woorde na willekeur saamgevoeg word om saamgestelde woorde te vorm. Daar is byna geen beperking op die aantal woorde wat in samestellings gebruik kan word nie, met die gevolg dat baie lang woorde gevorm kan word wat die uitleg van gedrukte teks onegalig maak indien woordafbreking nie aangewend word nie. Figuur 1.10 toon 'n ekstreme, kunsmatig geskepte geval van Duitse teks waarin woordafbreking nie gebruik is nie.

sogar länger als eine Zeile sein kann. Das Silbentrennungssystem SiSiSi ist zu empfehlen für Leitartikelverfasser bei Tageszeitungen, für Autoren schöngeistiger oder wissenschaftlicher Abhandlungen inhaltsreicher Wochenzeitschriften, für Aktiengesellschaftsvorstände zum Abfassen hoffnungsvoller Jahresergebnisvorausschauen, für Bundesverfassungsrichter bei Urteilsbegründungen und andere Schriftstückeerzeuger.

Figuur 1.10: Duitse teks sonder woordafbreking [Bar02]

Die mees sinvolle verdelingsposisies tydens woordafbreking is gewoonlik by die grense tussen die samestellende dele van saamgestelde woorde. Verder stem by lettergreepverdelingsposisies en die grense tussen die samestellende dele van saamgestelde woorde altyd ooreen [KS03].

In Tabel 1.3 word voorbeelde van lang saamgestelde woorde in verskillende tale getoon wat die belangrikheid en bruikbaarheid van die ontbinding van saamgestelde woorde illustreer. Dit maak die betekenis van die woorde en die korrelasie met Engels duidelik, wat belangrik is vir outomatiese linguistieke prosesse soos masjienvertaling [KAH08, PSN06, FF10], spraakherkenning [ADAL00] en inligtingherroeping [ABP08a].

Aangesien onbekende letterkombinasies dikwels geskep word tydens die woordsamestellingsproses, is dit vir die patroongebaseerde benadering wat ons volg uiters belangrik om die grense tussen die samestellende dele van saamgestelde woorde te bepaal.

1.3 Masjienleertegnieke

Masjienleer is daardie vertakking van kunsmatige intelligensie wat handel oor die studie en skep van stelsels wat uit data kan leer [Wik13g]. Hierdie leerproses (*afrigting*) bestaan daaruit dat parameters iteratief aangepas word na aanleiding van data wat verskaf word.

Afrigtingsdata vir 'n masjienleertegniek word uit voorbeelde van korrekte optrede vir 'n bepaalde taak gegenereer. By *gekontroleerde afrigting* bestaan afrigtingsdata uit *invoere* en hul ooreen-

Afrikaans	aandagafleibaarheidsindroomkonferensie aandag-afleibaarheid-sindroom-konferensie (*attention deficit syndrome conference*)
Duits	Höchstgeschwindigkeitsbegrenzung Höchst-geschwindigkeits-begrenzung (*maximum speed limit*)
Nederlands	ambtenarensalarisbespreking ambtenaren-salaris-bespreking (*public official salary discussion*)
Sweeds	rörelseuppskattningssökningsintervallsinställningar rörelse-uppskattnings-söknings-intervalls-inställningar (*motion estimation search range settings*)

Tabel 1.3: Lang woorde uit samestellende tale

stemmende uitvoere (*teikens*) wat uit verteenwoordigende data gegenereer is. Die tegniek word afgerig om vir elke invoer 'n ooreenstemmende uitvoer te lewer en die hoop bestaan dan dat die masjien sal kan *veralgemeen* om ook vir invoerdata wat nie deel van die afrigtingsdata was nie, korrekte uitvoere te lewer.

In hierdie studie rig ons die masjienleertegnieke *kunsmatige neurale netwerke*, *beslissingsbome* en Liang se woordafbrekingsalgoritme vir TEX af om lettergreep- en saamgesteldewoordverdeling te doen. Data vir so 'n afrigtingsproses word uit woorde wat korrek in lettergrepe of saamgestelde dele verdeel is, gegenereer.

- ▷ Kunsmatige neurale netwerke is deur die werking van biologiese senuweetelsels geïnspireer [Hag96]. Dit bestaan uit kunsmatige *neurone* (verwerkingseenhede) wat in lae gerangskik is. Die neurone in elke laag is volledig met dié in aangrensende lae verbind deur geweegde skakels. Invoer van buite, of vanaf ander neuronlae, word deur die neurone verwerk en na 'n volgende laag aangestuur om uiteindelik 'n uitvoer te lewer. Hierdie uitvoer word dan met die teikenuitvoer vergelyk en die gewigte op skakels word iteratief aangepas totdat die uitvoer genoegsaam met die teiken ooreenstem. 'n Afgerigte kunsmatige neurale netwerk kan dan gebruik word om die taak waarvoor dit afgerig is, doeltreffend uit te voer.

- ▷ Beslissingsbome is 'n masjienleertegniek waar voorbeelde van korrekte optrede gebruik word om 'n boomstruktuur te ontwikkel wat die uitkoms van elke item in 'n datastel deur middel van 'n reeks verdelings verteenwoordig [Nil96]. Beslissingsbome word dikwels vir klassifikasie gebruik in probleme waar die uitkoms 'n klas is waaraan 'n data-item behoort. 'n Klassifikasie-beslissingsboom word vir 'n datastel ontwikkel deur die data rekursief te verdeel sodat die onsekerheid oor die klas waaraan 'n data-item behoort, afneem soos in die boom afbeweeg word. By elke vertakking word besluit watter eienskap getoets moet word en wat die toets moet wees. Die volgorde van toetse word sodanig gekies dat die onsekerheid oor die klas waaraan 'n data-item behoort so gou as moontlik afneem.

- ▷ Frank Liang [Lia83] het sy woordafbrekingsalgoritme spesifiek vir TEX ontwikkel. Tydens 'n afrigtingsproses word taalspesifieke patrone uit 'n lys woorde wat korrek in lettegrepe verdeel is, gegenereer. Hierdie patrone word dan gebruik om vir 'n woord wat nie op 'n reël inpas nie, die mees geskikte afbreekposisie te bepaal.

Die afrigting van masjienleertegnieke vereis groot hoeveelhede data wat die probleem onder beskouing verteenwoordig. Vir die woordafbrekingsprobleem is dit dus belangrik om soveel moontlik woorde wat in lettergrepe verdeel is, te versamel.

Ten einde masjienleertegnieke te evalueer en met mekaar te vergelyk, word die prestasiemaatstawwe *akkuraatheid*, *presisie*, *herroeping* en die *F-telling* gebruik. Sien Afdeling 3.4 vir besonderhede.

1.4 Uiteensetting van proefskrif

Die proefskrif bestaan uit nege hoofstukke en vier bylaes. Hoofstuk 1 (hierdie inleiding) bestaan uit 'n inleidende beskrywing van belangrike aspekte rakende die studie wat onderneem is. Hoofstuk 2 handel oor aspekte rondom die versameling van data, die generering van afrigtings- en toetsdata en die skoonmaak van data. Interessanthede en onreëlmatighede rakende patrone wat met lettergreepverdeling gepaardgaan, word bespreek.

Hoofstuk 3 handel oor die verdeling van saamgestelde woorde. 'n Algoritme is ontwikkel om hierdie taak outomaties uit te voer deur stringpassing van die begin en einde van woorde af te doen, waarna die korrekte verdeling uit die samevoeging van begin- en eindwoorde volgens sekere kriteria bepaal word. Die algoritme word volledig bespreek en resultate op ons leksikon van $\pm 183\,000$ woorde word verskaf.

In Hoofstukke 4 tot 6 bespreek ons die verskillende masjienleertegnieke, naamlik kunsmatige neurale netwerke in Hoofstuk 4, beslissingsbome in Hoofstuk 5 en die TEX-algoritme in Hoofstuk 6. Vir elkeen word die basiese beginsels van die tegniek bespreek, hoe elkeen vir lettergreepverdeling en saamgesteldewoordverdeling afgerig is asook die resultate van elk. Verskille tussen die tegnieke word in Hoofstuk 7 uitgelig, die resultate van die drie tegnieke word vergelyk en gevolgtrekkings word gemaak.

In Hoofstuk 8 bespreek ons die proses wat gebruik is om 'n gekombineerde algoritme vir lettergreepverdeling te ontwikkel. Verskillende kombinasies van die optimale masjienleermodelle wat ontwikkel is, is getoets om so die mees effektiewe weergawe te bepaal. Die optimale algoritme is op teks uit koerante, tydskrifte, romans, ensovoorts getoets en die resultate word bespreek. Laastens verskaf ons 'n bespreking van die studie en idees vir verdere navorsing in Hoofstuk 9.

Bylae A handel oor probleme wat met Matlab ondervind is en ons oplossings daarvoor. In Bylae B verskaf ons besonderhede oor die grafiese uitvoer van die beslissingsboomsagteware CART, met verwysing na die finale beslissingsboom vir lettergreepverdeling. Bylae C bevat 'n bespreking van die datastruktuur wat vir die TEX-patrone ontwikkel is en in Bylae D word die bronkode van Perl-programme wat ontwikkel is, verskaf.

Hoofstuk 2

Data

Masjienleertegnieke benodig voorbeelde van korrekte gedrag vir afrigting. Dit is dus nodig om vir die woordafbrekingstaak soveel moontlik Afrikaanse woorde wat in lettergrepe verdeel is, te versamel. Ook vir die taak om saamgestelde woorde in hul samestellende dele te verdeel, moet voorbeelde van korrekte gedrag beskikbaar wees. Vir die afrigting van masjienleertegnieke is dit uiters belangrik om die data so skoon as moontlik te kry sodat die tegnieke nie verwar word deur teenstrydige gedrag nie.

In hierdie hoofstuk beskryf ons die proses wat gevolg is om 'n leksikon van Afrikaanse woorde uit 'n groot korpus te genereer en dit in lettergrepe te verdeel. Ons bespreek ook onreëlmatighede rakende lettergreepverdeling wat in en tussen woorde voorkom asook die invloedreikwydte van letters wat by sulke onreëlmatighede betrokke is. Daarna bespreek ons aspekte van die proses wat gevolg is om saamgestelde woorde in hul samestellende dele te verdeel. Laastens bespreek ons ook die onleding wat ons gedoen het van letterpatrone wat in woorde en ook rondom lettergreepverdelings voorkom.

2.1 Skep van leksikon

'n Lys van Afrikaanse woorde wat soveel as moontlik saamgestelde woorde insluit – ons leksikon – is saamgestel. Die volgende bronne is gebruik om teks in elektroniese formaat te bekom:

- ▷ Elektroniese woordeboeke soos die elektroniese HAT;
- ▷ Elektroniese woordelyste vanaf die internet;
- ▷ Elektroniese teks van bekende Afrikaanse boeke en koerante;
- ▷ Internetpublikasies en -nuusbriewe.

Uit hierdie groot korpus is 'n lys van woorde – ons leksikon – het ons 'n lys van die woorde wat daarin voorkom, gegenereer. Slegs 'n enkele weergawe van elke woord is in ons leksikon opgeneem, aangesien die frekwensie waarmee woorde in die korpus voorkom, nie in hierdie studie in aanmerking geneem word nie. (Program D.1.1.)

Ten einde anderstalige woorde uit ons leksikon te verwyder, is Engelse, Franse, Nederlandse en Duitse woordelyste van die internet bekom en met ons leksikon vergelyk. Ons het aanvaar dat

woorde wat slegs in ons leksikon voorgekom het wel Afrikaanse woorde is en dat woorde uniek aan die anderstalige lys, nie Afrikaanse woorde is nie. Woorde wat in beide ons leksikon en, byvoorbeeld, die Engelse lys voorgekom het, is per hand deurgegaan om woorde wat in beide Afrikaans en Engels geldig is, soos *word*, *stand*, *program*, te ondervang. (Program D.6.1.)

Nederlandse en Duitse woorde is verder verwyder deur kenmerkende eienskappe te gebruik, soos *tje*, *ouw*, *ooven* en *ijk* vir Nederlands en *sch* en *chen* vir Duits.

Aanvanklik is woorde wat met hoofletters geskryf word, hoofsaaklik eiename, in ons leksikon opgeneem. Dis egter moeilik, en soms onmoontlik om rekenaarmatig tussen eiename en woorde wat aan die begin van 'n sin met 'n hoofletter verskyn, te onderskei. Aangesien 'n woord nie anders afgebreek word wanneer dit met 'n hoofletter geskryf word nie, het ons besluit om alle woorde, eiename ingesluit, met kleinletters in ons leksikon op te neem.

In Afrikaans word beklemtoning aangedui deur aksenttekens op klinkers in woorde te plaas, byvoorbeeld *assebliéf*, *verál*, *léwe*, *lóóp*. 'n Woord word nie anders as gewoonlik afgebreek wanneer dit beklemtoon word nie. Ons het dus net die geaksentueerde woorde wat in ons korpus voorgekom het in ons leksikon opgeneem en het nie probeer om alle moontlike woorde wat geaksentueer kan word in te sluit nie.

2.2 Lettergreepverdeling

Om die woorde in ons leksikon in lettergrepe te verdeel, het ons die kunsmatige neurale netwerk wat in 'n vorige studie [Fic03] ontwikkel is, as eerste iterasie gebruik. Hierdie kunsmatige neurale netwerk is met relatief min data (±5 000 woorde) afgerig en alhoewel ons besef het dat dit nie alle woorde reg sal verdeel nie, het dit die basiese verdelings gedoen waarop ons kon uitbrei.

Die leksikon is per hand deurgewerk en foutiewe lettergreepverdelings is reggestel. Ons het ook Perl-programme ontwikkel om onreëlmatighede programmaties te identifiseer. Ons beskou dit as 'n onreëlmatigheid wanneer 'n bepaalde letterpatroon wat in meer as een woord voorkom, verskillend in lettergrepe verdeel word.

- ▷ Die frekwensie van letter- en koppeltekenpatrone van verskillende lengtes in ons leksikon is bepaal.
- ▷ Waar die koppelteken in 'n bepaalde letterpatroon op verskillende plekke voorkom, soos *ar-tjie* en *art-jie*, is alle woorde met hierdie patroon uit ons leksikon onttrek, ondersoek en foute is reggestel.

Patrone wat slegs een keer in ons leksikon voorgekom het, het gewoonlik op óf 'n fout óf 'n uitsondering gedui. Foute is reggestel en uitsonderings is aangeteken. Uitsonderings was dikwels die gevolg van woorde of eiename wat hul oorsprong in 'n ander taal het, soos *Jür-gen*, *ren-dez-vous*, *bou-clé*.

Uiteindelik het ons 'n lys van ±183 000 woorde wat grootliks korrek in lettergrepe verdeel is, saamgestel.

2.2.1 Onreëlmatighede

Tydens die ontfouting van ons leksikon is verskeie soorte onreëlmatighede geïdentifiseer wat tot probleme met die afrigting van masjienleertegnieke en ook met rekenaarmatige woordafbreking kan lei.

▷ *Onreëlmatige verdeling van volledige woorde*

Soms word volledige woorde met presies dieselfde spelling op grond van betekenis verskillend in lettergrepe verdeel. Voorbeelde hiervan word in Tabel 2.1 getoon.

Woord	**Betekenis 1**	**Betekenis 2**
eselskop	e-sels-kop	e-sel-skop
geklik	ge-klik	gek-lik
kateter	ka-te-ter	kat-e-ter
proeslag	proe-slag	proes-lag
pronkertjie	pronk-ert-jie	pron-ker-tjie
sandaal	san-daal	sand-aal
trompop	trom-pop	tromp-op

Tabel 2.1: Ononderskeibare onreëlmatighede

Dit is moeilik, en dikwels onmoontlik, om sulke woorde rekenaarmatig korrek te verdeel, selfs al sou sintaktiese inligting beskikbaar wees. Byvoorbeeld, uit 'n sin soos *Sy loop op die strand en trap op 'n sandaal* is dit nie duidelik of daar na 'n san-daal (skoen) of 'n sand-aal (seedier) verwys word nie.

Dit is wel moontlik om in sekere gevalle hierdie soort probleme aan te spreek deur die konteks waarin die woord gebruik word in ag te neem, maar aangesien ons slegs letterpatrone gebruik, het ons nie hierdie opsie nie. 'n Uitsonderingslys sal ook nie in hierdie geval 'n oplossing bied nie, want geen onderskeiding kan tussen die moontlikhede gemaak word nie. Die enigste manier om hierdie probleem aan te spreek is om geen lettergreepverdeling toe te laat nie en die gebruiker tydens woordafbreking daarop attent te maak indien so 'n woord 'n afbrekingskandidaat is.

▷ *Onreëlmatige verdelings binne woorde*

Identiese gedeeltes van woorde (letterpatrone) word soms na gelang van betekenis, uitspraak of herkoms verskillend in lettergrepe verdeel. Voorbeelde van sulke onreëlmatighede word in Tabel 2.2 getoon.

In hierdie gevalle is dit dikwels moontlik om uit die nabye konteks (letters rondom die verdelingspunt) te bepaal waar lettergreepverdeling moet plaasvind. Indien die vensterwydte wat vir die afrigting van masjienleertegnieke gebruik word groot genoeg is, behoort sulke gevalle ondervang te word. (Sien Afdeling 4.2.1 vir besonderhede oor vensterwydtes.)

Woorddeel	**Weergawe 1**	**Weergawe 2**
dienste	ver-dien-ste	kerk-diens-te
ekst	eks-tro-vert	ek-sta-se
eum	li-no-le-um	mu-seum
gnos	ag-nos-ti-ci	gnos-ti-ci
komste	in-kom-ste	by-een-koms-te
sju	bro-sjure	dis-junk
tjie	kaart-jie	paar-tjie

Tabel 2.2: Onreëlmatighede in woorddele

▷ *Onreëlmatige verdelings rondom die letter* s

Onreëlmatighede kom dikwels voor waar letterkombinasies die letter *s* insluit. Faktore wat tot hierdie verskynsel bydra, is die verbindings-*s* wat by die vorming van saamgestelde woorde gebruik word, asook die opsionaliteit in woordafbrekingsreëls vir sulke letterkombinasies. Reël 1.8 in die AWS [Die09] lees soos volg:

> *Indien drie konsonantletters langs mekaar tussen vokaalletters staan en die eerste konsonantletter 'n s is, kan daar vóór of ná die s afgebreek word.*

Die opsionaliteit van hierdie reël het noodwendig onreëlmatigheid rondom die letter *s* tot gevolg wat probleme by die afrigting van masjienleertegnieke kan veroorsaak. Vir die woord *distrik*, byvoorbeeld, is beide *dis-trik* en *di-strik* aanvaarbaar.

In Tabel 2.3 word voorbeelde van onreëlmatighede as gevolg van die verbindings-*s* getoon.

beroep-sport	groeps-portret
edel-staal	handels-taal
garing-soort	bewarings-oord
gevoel-sintuig	gevoels-indruk
omgewing-sake	omgewings-aktiwiteit
voorsitter-stoel	voorsitters-toespraak

Tabel 2.3: Onreëlmatighede as gevolg van verbindings-*s*

Vergelyk byvoorbeeld die woorde *beroep-sport* en *groeps-portret*. Die onderstreepte gedeeltes is identies in die twee woorde, maar die verdeling verskil. Eers wanneer ons ten minste vyf letters links van die letter *s* of vyf letters regs daarvan beskou, is dit duidelik waar die woord verdeel moet word. Tensy die venstergrootte vir die afrigting van masjienleertegnieke gebruik word groot genoeg is om sulke gevalle te ondervang, sal dit nodig wees om probleemwoorde in 'n uitsonderingslys op te neem.

Die aantal letters links en/of regs van die letter *s* wat met sekerheid aandui waar die woord verdeel moet word, staan as die *invloedreikwydte* van *s* bekend. 'n Ontleding van invloedreikwydtes kan moontlik onreëlmatighede, en sodoende ook verwarring by outomatiese lettergreepverdeling beperk. Meer hieroor in Afdeling 2.2.2.

▷ *Onreëlmatighede tussen en in woordeboeke*

Woordeboeke is gebruik om lettergreepverdeling per hand te kontroleer. Ons het die HAT [OG09] en Pharos [Pha05] as bronne gebruik, aangesien beide lettergreepverdeling aandui. Tydens hierdie proses is onreëlmatighede binne, en ook tussen die twee woordeboeke opgemerk. In Tabel 2.4 word voorbeelde hiervan getoon.

HAT		**Pharos**	
	–	en-kou-stiek	en-kous-ties*
guns-te-ling	gun-stig	guns-tig	
(Is-ma-el)	Is-rael	Is-ra-el	Is-rae-liet
ma-er-heid/ver-ma-er	maer	ma-er	
(si-ne-re-se)	sin-er-gis-me	si-ner-gis-me	
	sin-es-te-sie	si-ne-ste-sie	
	ta-xi**	tax-i	
vors-tin	vor-ste-dom	vors-te-dom	vor-ste-lik
(in teks: wor-stel)	wors-tel	wor-stel	

Tabel 2.4: Onreëlmatighede binne en tussen woordeboeke

* Hierdie onreëlmatigheid is daaraan te wyte dat die klem op verskillende plekke in die woorde val, naamlik *en-kou-stiek* en *en-kous-ties*. Ons beskou dit as 'n uitsonderlike geval.

** Volgens Reël 1.16 in die AWS moet verdeling ná die x plaasvind.

Let daarop dat die woordeboeke die hoofklem in woorde verskillend aandui: Pharos onderstreep die klinker of klinkergroep wat die hoofklem kry, terwyl die HAT 'n apostroof gebruik om aan te toon dat die voorafgaande lettergreep die hoofklem kry. In beide woordeboeke word lettergreepverdeling deur 'n verhewe punt aangedui, terwyl die apostroof by die HAT ook lettergreepverdeling aandui. In die tabel hierbo gebruik ons lettergreepverdeling in ooreenstemming met die bronne, maar dui dit deurgaans met 'n koppelteken aan.

2.2.2 Invloedreikwydte

Onreëlmatigheid ten opsigte van lettergreepverdeling het meer as een oorsaak. Ons het reeds die geval bespreek wat voorkom as gevolg van die verbindings-*s*. (Sien Tabel 2.3.)

Die samestelling van woorde wat toevallig dieselfde letterkombinasie rondom die verdelingspunte het, lewer soms teenstrydige lettergreepverdeling in die posisie waar die woorde in samestellende dele verdeel word. Voorbeelde hiervan is *agter-ente* teenoor *vaste-rente* en *verstek-waarde* teenoor *kleinste-kwadrate*.

Om die rekenaar in staat te stel om tussen sulke letterkombinasies te onderskei, moet die venster wat by masjienleer vir patroongenerasie gebruik word die letters insluit wat met sekerheid aandui waar die woord verdeel moet word.

Die aantal letters links en/of regs van die letter wat by die onreëlmatigheid betrokke is, soos die *r* in *vasterente* en *agterente* en die *s* in *doodskreet* en *doodskree*, wat met sekerheid aandui waar afbreking moet plaasvind, staan as die *invloedreikwydte* van die letter bekend. Wanneer die woordegedeeltes wat by die onreëlmatigheid betrokke is met woordgrense ooreenstem, strek die invloedreikwydte tot by die witspasie wat die begin/einde van die woord aandui.

In Tabel 2.5 word voorbeelde van invloedreikwydte getoon. Die onderstreepte gedeelte is telkens identies in die onderskeie woorde, terwyl die onreëlmatige letter vet gedruk word. Die aantal letters wat links en regs van hierdie letter beskou moet word om die onreëlmatigheid te ondervang, word onderskeidelik in die kolomme aan die linker- en regterkant getoon. So is 'n 3–5-vensterkonfigurasie, byvoorbeeld, nodig om die onreëlmatigheid in *agterente* en *vasterente* te ondervang en 'n 4–3-konfigurasie vir *verstekwaarde* en *kleinstekwadrate*.

Links	**Woord 1**	**Woord 2**	**Regs**
3	agte**r**-ente	vaste-**r**ente	5
4	verste**k**-waarde	kleinste-**k**wadrate	3
5	diens-teraad	verdien-**s**tereëling	4
5	rooste**r**-oond	testoste-**r**oonvlak	4
6	karaktereie**n**-aardigheid	trekkereie-**n**aar	4
7	vervoerdiens-te	verdien-**s**te	3
8	regerings-taal	legering-**s**taal	5
5	doods-kreet	dood-**s**kree	5

Tabel 2.5: Invloedreikwydte links en regs van onreëlmatige letter

Hieruit blyk dit dat 'n vensterkonfigurasie van ten minste 8–5 nodig sal wees om ál hierdie onreëlmatighede te ondervang. 'n Ander moontlikheid is om 'n uitsonderingslys te gebruik, aangesien groot invloedreikwydtes gewoonlik slegs in uitsonderlike gevalle voorkom.

2.3 Saamgesteldewoordverdeling

In Afrikaans word saamgestelde woorde uit 'n arbitrêre aantal bestaande woorde geskep wat aanmekaar geskryf word. In hierdie samestellingsproses word dikwels nuwe letterkombinasies geskep wat tot afbrekingsfoute by die grense tussen die samestellende dele van saamgestelde woorde kan lei. Ons het dus besluit om 'n weergawe van ons leksikon te skep waarin saamgestelde woorde in hul samestellende dele verdeel is.

Ten einde die verdelingsproses te formaliseer, het ons 'n algoritme ontwikkel wat die taak outomaties uitvoer (Hoofstuk 3). In hierdie ontwikkelingsproses moes ons voortdurend besluite neem oor wat as samestellende dele beskou behoort te word.

Ons moes besluit of (en watter) voor- en/of agtervoegsels as samestellende dele beskou moet word en van die res van die woord afgeskei behoort te word. Ons het hierdie besluite geneem met woordafbreking (of lettergreepverdeling) as hoofdoel en nie streng volgens taalreëls nie.

Sommige voorvoegsels, soos *af*, *op*, *deur*, ens. is woorde in eie reg, maar dis nie altyd duidelik of dit as 'n samestellende deel van 'n woord beskou kan word nie. In *afbreek* en *afkyk*, byvoorbeeld, sou *af* as 'n samestellende deel beskou kan word, maar in *afbraakproduk*, byvoorbeeld, beskou ons *afbraak* as 'n enkelvoudige woord. Netso is ons nie seker of 'n woord soos *deurmekaar* as 'n saamgestelde woord beskou kan word nie, maar daar is nie twyfel oor *deurblaai* en *deurloop* nie. Tabel 2.6 toon verdere voorbeelde van sulke probleemgevalle.

Voor-	**Verdeel**		**Voor-**	**Verdeel**	
voegsel	**Seker**	**Onseker**	**voegsel**	**Seker**	**Onseker**
aan	aansmeer	aanval	in	inspuit	instink
	aandra	aanwesig		inspring	inhoud
af	afskryf	afslag	her	hereksamen	herhaal
	aftrek	afskeid		herbeplan	herstel
by	byangel	bybehore	mis	misbaksel	misbruik
	bysteek	byvoeg		misgeboorte	misdaad
deur	deurblaai	deurmekaar	on	onakkuraat	onnosel
	deurswem	deurbring		ongedateer	onheil

Tabel 2.6: Onsekerheid by voorvoegsels

As gevolg van hierdie onsekerheid het ons dus besluit om voorvoegsels nie af te skei nie. Ons neem aan dat hierdie verdelingsposisies deur lettergreepverdelingstegnieke tydens woordafbreking bepaal sal word. Slegs wanneer 'n woord wat as voorvoegsel gebruik word 'n selfstandige naamwoord is, soos in *oor-skulp*, *deur-knop*, *mis-wolke*, ensovoorts, word sulke woorde afgeskei. Saamgestelde woorde wat voorvoegselsamestellings soos *agteruit* en *vooraf* bevat, is egter wel verdeel as *agteruit-gaan* en *vooraf-beplan*.

Die meeste agtervoegsels is nie volwaardige woorde nie, byvoorbeeld *-ing*, *-tjie*, *-jie*, en dit word nie afgeskei nie. Daarenteen word die agtervoegsel *-agtig* en die verbuigings daarvan wel afgeskei aangesien dit met 'n klinker begin en dikwels tydens lettergreepverdeling by foute betrokke is. (Ons doen dit sodat masjienleertegnieke hierdie gedrag kan aanleer.) Ander lang agtervoegsels wat ook afgeskei word, is *-sinnig*, *-sugtig*, en hul verbuigings.

Tegniese woorde soos *geografies*, *elektroskoop*, *fisionomie* bestaan uit 'n samestelling van twee *pseudowoorde* – beide dra betekenis, maar is nie volwaardige woorde wat alleen gebruik word nie. Die pseudowoorde aan die begin (*geo*, *elektro*, *fisio*) word soos voorvoegsels gebruik en staan as *tegnostamme* bekend. Die pseudowoorde aan die einde (*grafies*, *skoop*, *nomie*) kom nie as selfstandige woorde voor nie en word dus nie van die tegnostamme geskei nie. (Die woord *grafies* kom wel in ons leksikon voor, maar het 'n ander betekenis.)

In gevalle waar tegnostamme egter wel saam met volwaardige woorde voorkom, soos in *biodiversiteit*, *fisioterapeut*, *astrofisikus*, word die tegnostam wel van die res van die woord afgeskei sodat *telekommunikasie*, byvoorbeeld, as *tele-kommunikasie* verdeel word.

Die weergawe van ons leksikon waarin saamgestelde woorde in hul samestellende dele verdeel is, bestaan uit 43% enkelvoudige woorde en 57% saamgestelde woorde. Byna 90% van die

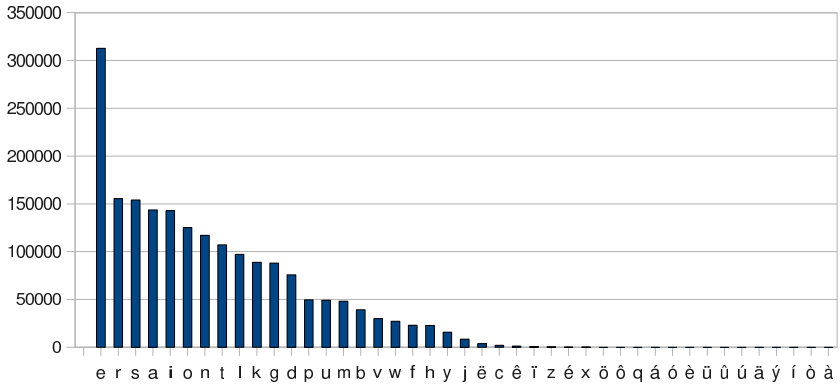
saamgestelde woorde bestaan uit twee samestellende dele, 9% uit drie dele, 0,5% uit vier dele, 0,02% uit vyf dele en slegs twee voorde bestaan uit ses dele, naamlik *motor-werk-tuig-herstel-werk-dienste* en *streek-storie-saam-sing-musiek-aand*.

2.4 Patroonontleding

Ten einde 'n beter begrip te kry van die patrone wat letters in Afrikaanse woorde vorm, het ons letter- en lettergreepverdelingspatrone van verskillende lengtes ontleed. Ons bespreek vervolgens die ontleding van letterpatrone in woorde en daarna die ontleding van patrone wat met met lettergreepverdeling gepaardgaan.

2.4.1 Letterpatrone

In Afrikaans word 43 letters gebruik, naamlik die 26 letters van die alfabet en 17 klinkers met diakritiese tekens, soos ë, ê, ó, ensovoorts. In Figuur 2.1 word die frekwensie-ontleding van hierdie letters soos dit in ons leksikon voorkom, getoon. Let daarop dat die frekwensie waarmee 'n letter of patroon in die leksikon voorkom in Figure 2.1 tot 2.8 op die vertikale as getoon word.

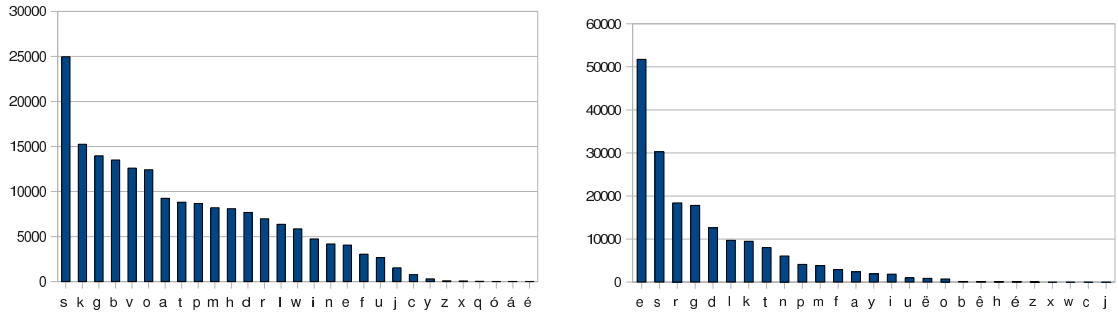


Figuur 2.1: Letters wat in Afrikaanse woorde voorkom

Die klinker *e* het die hoogste frekwensie (16,2% van die totale aantal letters in die woordelys) terwyl die medeklinkers *r* en *s* die meeste voorkom. Die letters *z*, *x* en *q*, asook die letters met diakritiese tekens soos *û*, *ä* en *ò* kom relatief min voor.

In Figuur 2.2 word die frekwensieverdeling van letters getoon wat aan die begin en einde van woorde voorkom. Die frekwensie van woorde wat met die letter *s* begin is die hoogste, terwyl woorde ook baie dikwels met *k*, *g*, *b*, *v* en *o* begin. Die frekwensie van woorde wat met 'n *e* eindig is weer die hoogste, terwyl woorde wat met 'n *s* eindig, die tweede hoogste frekwensie het.

Dis belangrik om daarop te let dat 13,6% van woorde met die letter *s* begin terwyl 16,3% van woorde met die letter *s* eindig. Dit is dus duidelik dat die letter *s* probleme by saamgestelde woorde kan veroorsaak – veral as ons ook die verbindings-*s* in gedagte hou. In teenstelling



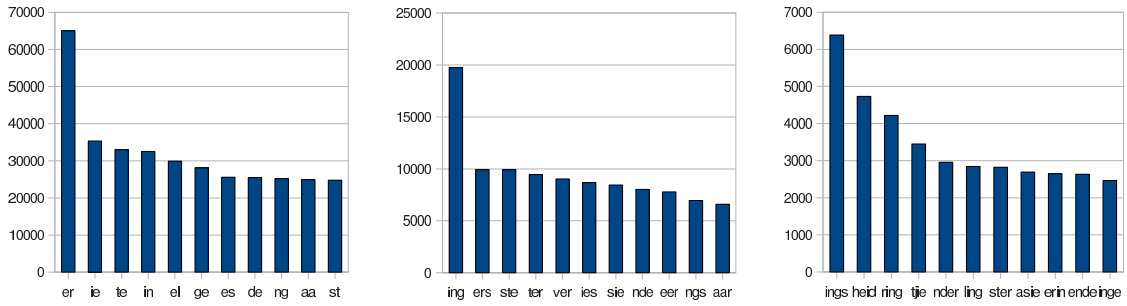
(a) Begin van woorde (b) Einde van woorde

Figuur 2.2: Letters waarmee woorde begin en eindig

hiermee kom die letter *b*, byvoorbeeld, baie dikwels aan die begin van woorde voor, maar byna nooit aan die einde van woorde nie.

Letterpatrone (sonder koppeltekens) van lengtes twee tot agt is uit ons leksikon onttrek. In Figuur 2.3 word die frekwensieverdelings van onderskeidelik twee-, drie- en vierletterpatrone getoon. (Vir patrone wat uit twee of meer letters bestaan, sluit ons voortaan net die patrone met die hoogste frekwensie in.)

Die tweeletterpatroon *er* het die hoogste frekwensie – dit kom ±65 000 keer in die ±183 000 woorde van ons leksikon voor. Die drieletterpatroon *ing* kom die meeste voor (byna 20 000 keer), wat ook duidelik is uit die ontleding van vierletterpatrone, waar *ings, ring, ling* en *inge* onder die patrone met die hoogste frekwensie voorkom.



(a) Tweeletterpatrone (b) Drieletterpatrone (c) Vierletterpatrone

Figuur 2.3: Frekwensieverdeling van letterpatrone

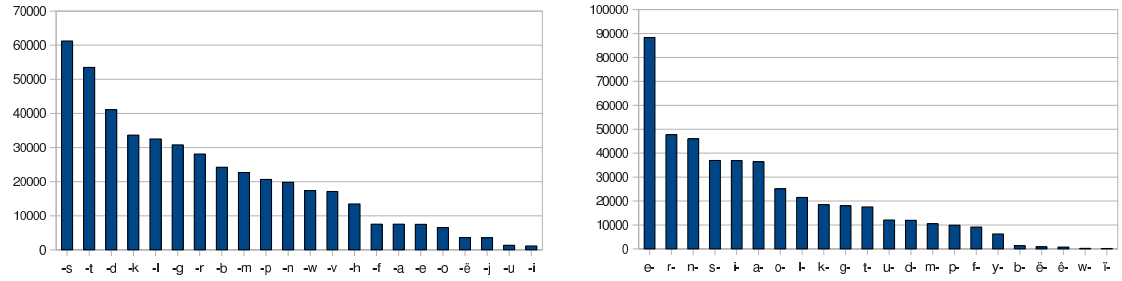
2.4.2 Lettergreeppatrone

Aangesien ons in die patrone van letters rondom verdelingspunte belangstel, het ons ook hierdie patrone op verskillende maniere ondersoek. Ons het die patrone van letters aan die begin en einde van lettergrepe beskou asook die frekwensie van volledige lettergrepe binne woorde en aan die begin en die einde van woorde. Laastens het ons ook die frekwensie van verskillende lettergreeplengtes ondersoek.

▷ *Letterpatrone aan begin en einde van lettergrepe*

Die frekwensieverdeling van verskillende letterpatrone waarmee lettergrepe *binne woorde* begin en eindig, is bepaal. Hier beskou ons dus nie lettergrepe aan die begin en einde van woorde nie.

Hierdie patrone word daaraan uitgeken dat letters aan die begin van sulke lettergrepe ná 'n koppelteken staan en aan die einde van lettergrepe vóór 'n koppelteken. Byvoorbeeld, die begin van die tweede lettergreep in *ver-deel* word aangedui deur *-d* en die einde van die eerste lettergreep deur *r-*. In Figuur 2.4 word die frekwensie van enkelletters aan die einde en aan die begin van lettergrepe getoon.

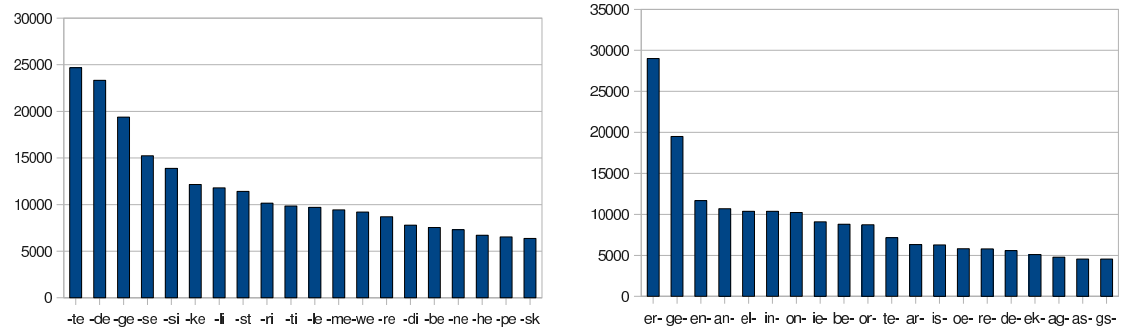


(a) Begin van lettergrepe (b) Einde van lettergrepe

Figuur 2.4: Letters aan begin en einde van lettergrepe

Soos by die frekwensies van letters aan die begin en einde van woorde sien ons dat lettergrepe baie dikwels met 'n *e* eindig, maar selde daarmee begin. Die letter *s* kom egter aan die begin sowel as aan die einde van lettergrepe met hoë frekwensie voor wat daarop dui dat probleme met verdeling rondom die letter *s* verwag kan word.

In Figuur 2.5 word die frekwensieverdeling van tweeletterpatrone aan die begin en einde van lettergrepe getoon. Soos voorheen staan letterpatrone aan die einde van lettergrepe vóór die koppelteken staan en aan die begin van lettergrepe ná die koppelteken.



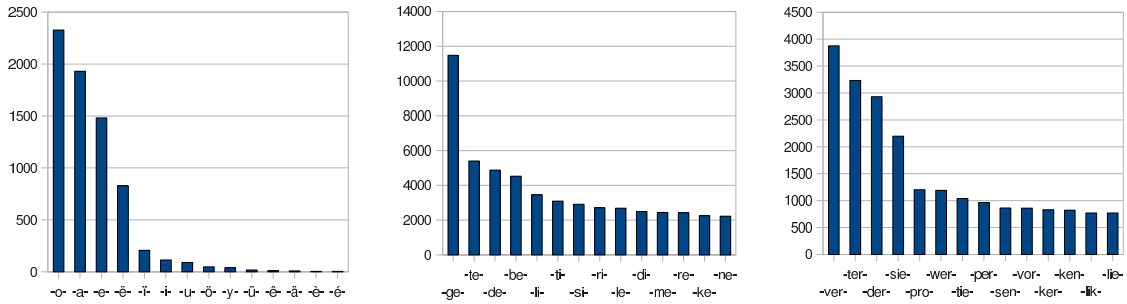
(a) Begin van lettergrepe (b) Einde van lettergrepe

Figuur 2.5: Tweeletterpatrone aan begin en einde van lettergrepe

Die tweeletterpatroon *er* kom dus die meeste aan die einde van lettergrepe voor, terwyl die patroon *te* die meeste aan die begin van lettergrepe voorkom.

▷ *Volledige lettergrepe*

Figuur 2.6 toon frekwensieverdelings van volledige lettergrepe wat binne woorde (tussen koppeltekens) voorkom. So is *-ter-*, byvoorbeeld, die volledige lettergreep binne die woord *let-ter-greep*.



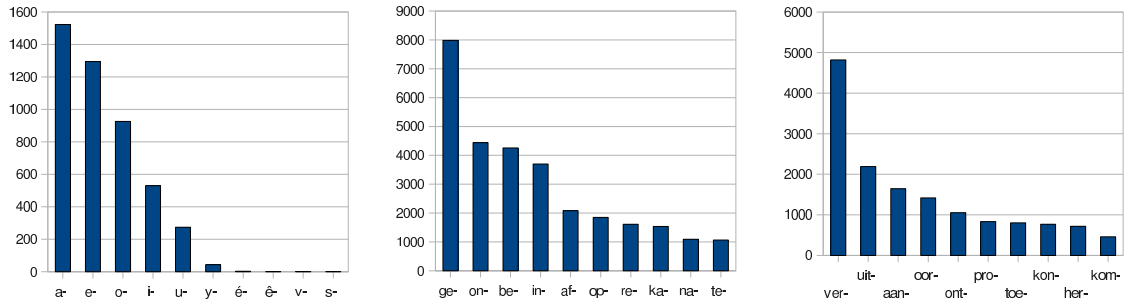
(a) Enkelletters (b) Twee letters (c) Drie letters

Figuur 2.6: Volledige lettergrepe binne woorde

Die enkelletter wat die meeste as 'n volledige lettergreep binne woorde voorkom, is *o* wat in woorde soos *bi-bli-o-teek* en *pe-ri-o-de* voorkom. Die lettergreep *-a-* kom voor in woorde soos *te-a-ter* en *si-tu-a-sie*.

Die tweeletterpatroon wat meeste as volledige lettergreep binne woorde voorkom, is *-ge-* wat in woorde soos *om-ge-wing* en *drei-ge-men-te* voorkom, asook in samestellings soos *boek-ge-leer-des* en *fyn-ge-kap*. Die drie-, vier-, vyf- en sesletterpatrone wat die meeste voorkom, is onderskeidelik *ver* soos in *spits-ver-keer*, *stel* in *spier-stel-sel*, *heids* in *snel-heids-grens*, *stuurs* in *be-stuurs-plan* en *straats* in *ma-gi-straats-kan-toor*.

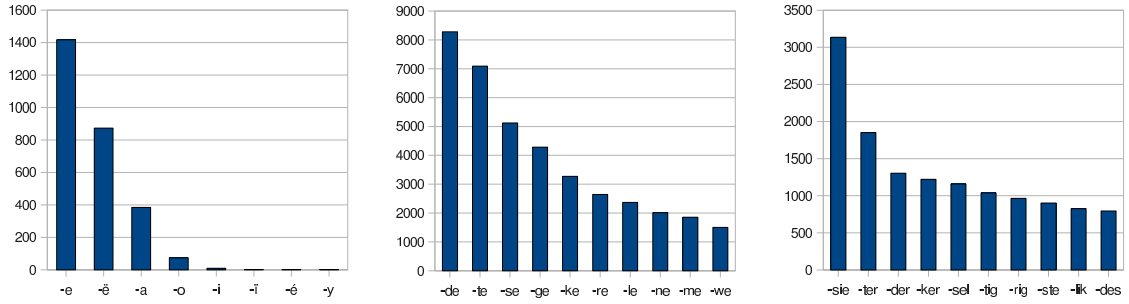
Die frekwensieverdelings van volledige lettergrepe aan die begin en einde van woorde word in Figure 2.7 en 2.8 getoon.



(a) Enkelletters (b) Twee letters (c) Drie letters

Figuur 2.7: Lettergrepe aan die begin van woorde

Die letter *a* kom meeste aan die begin van 'n woord as 'n lettergreep voor, en wel in woorde soos *a-fri-kaans*, *a-de-laar*, *a-dres*, ens. Dit is te verwagte dat die tweeletter-lettergreep *ge* die hoogste frekwensie aan die begin van woorde sal hê, aangesien die verledetyd van werkwoorde daarmee begin.



(a) Enkelletters (b) Twee letters (c) Drie letters

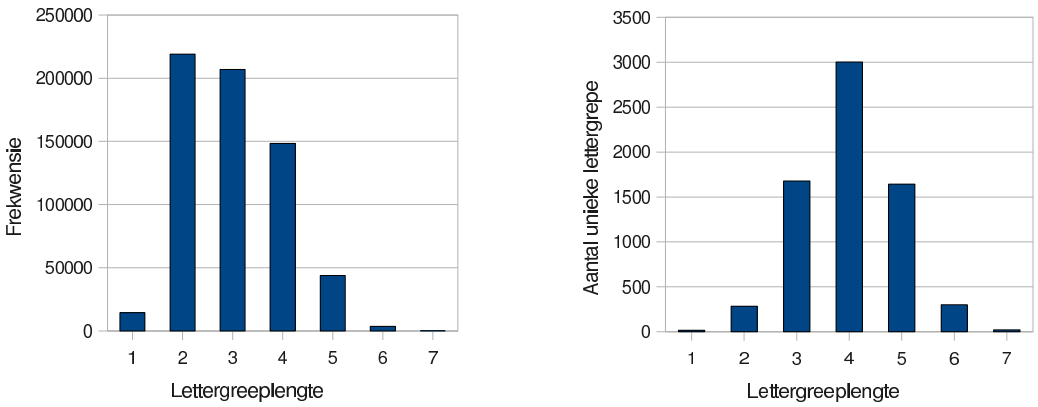
Figuur 2.8: Lettergrepe aan die einde van woorde

Ook die hoë frekwensie van die letters *e* en *ë* as lettergrepe aan die einde van woorde is te verwagte, aangesien dit gebruik word in die meervoudsvorm van woorde wat op 'n klinker eindig, soos *krui-e*, *lan-de-ry-e*, *or-gi-de-ë*, ens. Wanneer 'n woord met 'n medeklinker eindig, word die medeklinker na die volgende lettergreep oorgedra, wat die hoë voorkoms van tweeletterlettergrepe wat op 'n *e* eindig, verklaar. Voorbeelde hiervan is *tan-de*, *har-te*, ens.

Lettergreeplengte

'n Ontleding van die lengte van lettergrepe in ons leksikon het die resultate in Figuur 2.9 gelewer. Die frekwensie waarmee lettergrepe van verskillende lengtes voorkom, word in Figuur 2.9(a) getoon. Hieruit is dit duidelik dat kort lettergrepe – twee en drie letters lank – die meeste voorkom.

Die aantal unieke lettergrepe vir verskillende lengtes word in Figuur 2.9(b) getoon. Hieruit is dit duidelik dat die grootste verskeidenheid voorkom by lettergrepe wat uit vier letters bestaan, terwyl die aantal unieke lettergrepe wat uit drie en vyf letters bestaan byna dieselfde is. By masjienleer waar besluit moet word oor die venstergrootte wat vir patroongenerasie gebruik word, kan hierdie inligting 'n waardevolle bydrae lewer.



(a) Lettergreepfrekwensie (b) Unieke lettergrepe

Figuur 2.9: Lettergreepfrekwensie

Hoofstuk 3

Saamgesteldewoordverdeling: 'n rekursiewe algoritme

In hierdie hoofstuk bespreek ons die ontwikkeling van 'n algoritme wat ontwerp is om die grense tussen die samestellende dele van saamgestelde woorde outomaties te bepaal. Hierdie algoritme gebruik die woorde in ons leksikon om die grense tussen samestellende dele in die woorde van dieselfde leksikon te bepaal.

3.1 Inleiding

Die hoofkenmerk van samestellende tale soos Afrikaans, Duits, Nederlands, ens. is dat saamgestelde woorde gevorm word deur bestaande woorde saam te voeg. Hierdie samestelling het dikwels lang woorde tot gevolg, soos *tweedehandsemotorverkoopsmannevakbondstakingsvergadering*, *overeenstemmingsbeoordelingsprocedures* (Nederlands), en *siebenhundertsiebenundsiebzigtausendsiebenhundertsiebenundsiebzig* (777 777 in Duits). Vir linguistieke prosesse soos masjienvertaling [KAH08, PSN06, FF10], spraakherkenning [ADAL00] en inligtingherroeping [ABP08a] is dit nodig om die grense tussen die dele waaruit sulke woorde saamgestel is, te bepaal.

Ten einde die woorde in ons leksikon in hul samestellende dele te verdeel, het ons 'n rekursiewe algoritme (die SG-algoritme) ontwikkel wat deur middel van volledige stringpassing van die begin en einde van woorde af die grense tussen samestellende dele bepaal.

In die volgende afdelings bespreek ons die beginsels wat gebruik is om die algoritme te ontwikkel, asook die resultate wat behaal is met die toepassing van die finale algoritme op ons volledige leksikon.

3.1.1 Vorige navorsing

Bestaande tegnieke wat vir die verdeling van saamgestelde woorde ontwikkel is, is hoofsaaklik op woordfrekwensie, waarskynlikheid en stringpassing gebaseer – dus hoofsaaklik korpusgebaseerde tegnieke. Voorbeelde van sulke tegnieke is die volgende:

- ▷ Schiller [Sch05] beskryf 'n eindige-toestandsamesteller (*finite-state compiler*) wat op geweegde samestellingsegmente gebaseer is. Dit gee voorkeur aan verdelings met die minste segmente, asook segmente met die hoogste frekwensie in 'n afrigtingsdatastel.

- ▷ 'n Saamgestelde-woordverdeler vir Duits is deur Alfonseneca, Bilac en Pharies [ABP08b] ontwikkel wat verskillende gewigtoekenningsmetodes gebruik, naamlik frekwensie- en waarskynlikheidsgebaseerde metodes, onderlinge inligting, teenwoordigheid in ankertekste en 'n ondersteuningsvektormasjien- (*Support vector machine*) klassifiseerder.

- ▷ Philipp Koehn en Kevin Knight [KK03] het empiriese metodes vir die verdeling van saamgestelde woorde ondersoek. Ten einde saamgestelde woorde met bekende woorde en verbindingsmorfeme te dek, het hulle 'n volledige rekursiewe soektog in 'n matriks van bekende woorde uitgevoer. Hulle het die aanname gemaak dat hoe meer gereeld 'n woord in 'n korpus voorkom, hoe groter is die kans dat dit deel van 'n saamgestelde woord sal uitmaak. Hulle het tot die gevolgtrekking gekom dat saamgestelde woorde tot *yl-dataprobleme* lei en dat Duits-na-Engels-vertaling verbeter wanneer saamgestelde woorde verdeel word.

- ▷ Gabriele Kodydek [Kod00] het 'n rekursiewe woordverdelingsalgoritme as deel van 'n woordontledingstelsel vir Duits ontwikkel. Dit verdeel woorde in hul betekenisvolle dele (stamme en voor- en agtervoegsels) volgens die reëls vir verbuiging, afleiding en samestelling. Dit word in die SiSiSi-stelsel gebruik vir betroubare, betekenisdraende woordafbreking, voltekssoektogte en ook in 'n beperkte mate vir speltoetsing.

- ▷ Monz en De Rijke [MDR02] het 'n rekursiewe ontbindingsprosedure gebruik wat gulsig die kortste deelstring van links en regs van 'n woord uit 'n leksikon kies. Dit is egter nie in staat om woorde met verskeie moontlike verdelings te hanteer nie.

- ▷ 'n Vorm van langstestringpassing (LSP) is deur Brown [Bro02] toegepas om ooreenstemmende woorde in Duits en Engels in 'n mediese omgewing te identifiseer en dit te gebruik om saamgestelde woorde te verdeel. Hy het bepaal dat korpusgedrewe ontbinding die prestasie van 'n voorbeeldgebaseerde masjienvertalingstelsel verbeter het.

- ▷ Maja Popović, *et al.* [PSN06] het bevind dat linguistiese en korpusgebaseerde saamgesteldewoordverdeling sowel as verbeterde woordoplyning masjienvertaling tussen Engels en Duits verbeter het.

- ▷ Pilon, *et al.* [PPvH08] het onder andere ook LSP vir saamgesteldewoordverdeling in Afrikaans gebruik. Ooreenstemmende langste stringe van begin en einde van woorde is gebruik, maar hul resultate was teleurstellend.

Alfonseca, *et al.* [ABP08a] het daarop gewys dat frekwensie- en waarskynlikheidgebaseerde metodes negatief beïnvloed word deur die toevallige hoë frekwensie van woorde in 'n korpus en die voorkeur vir die minimum aantal samestellende dele.

3.2 Ontwikkeling van algoritme

Aangesien saamgestelde woorde geskep word deur bestaande woorde saam te voeg, maak dit sin om 'n lys van bestaande woorde te gebruik om woorde te ontbind. Hierdie algoritme gebruik volledige stringpassing vir die ontbindingstaak. In teenstelling met konvensionele LSP waar die *langste* stringe van óf die begin, óf die einde van die woord af gebruik word, gebruik ons *alle* gemeenskaplike stringe vanaf die begin en einde van woorde in die ontbindingsproses.

3.2.1 Basiese beginsels

Die leksikon van ±183 000 woorde word rekursief as verwysingslys (VL) gebruik om woorde in die leksikon self in samestellende dele te verdeel.

Vir 'n woord van lengte wl word stringe van lengtes 2, 3, ..., $wl - 2$ vanaf die begin en einde van 'n woord onderskeidelik met VL vergelyk en ooreenstemmende woorde word onttrek. Hierna word die woorde wat van die begin van die woord af onttrek is onderling met woorde wat van die einde van die woord af onttrek is, gekombineer. Slegs wanneer die lengte van so 'n kombinasie met wl ooreenstem, beskou ons dit as die korrekte verdeling.

Beskou byvoorbeeld die woord *woordeboek* met $wl = 10$. Stringe van lengtes 2, 3, ..., 8 van die begin van die woord af en ook stringe van lengte 2, 3, ..., 8 van die einde van die woord af word een-vir-een met VL vergelyk. Daar word dus 14 keer deur VL gegaan om ooreenstemmende woorde te onttrek. In Tabel 3.1 word hierdie proses geïllustreer.

Lopie deur VL	**String vanaf die woord se**		**Lopie deur VL**
	begin	**einde**	
1	wo	**ek**	8
2	woo	oek	9
3	woor	**boek**	10
4	**woord**	eboek	11
5	**woorde**	deboek	12
6	woordeb	rdeboek	13
7	woordebo	ordeboek	14

Tabel 3.1: Stringe van begin en einde van woord met VL vergelyk

Die geïdentifiseerde woorde van die begin van die woord af (*woord* en *woorde*) word in 'n *beginwoordelys* (BW) geplaas en dié van die einde af (*ek* en *boek*) in 'n *eindwoordelys* (EW). Ons het dus die volgende:

BW	EW
woord	ek
woorde	boek

Om nou te bepaal of 'n woord verdeelbaar is, word woorde in BW onderling met woorde in EW gekombineer. In hierdie geval het ons die volgende:

Kombinasie	Gekombineerde lengte
woord-ek	$kl = 7 < wl$
woorde-ek	$kl = 8 < wl$
woord-boek	$kl = 9 < wl$
woorde-boek	$kl = 10 = wl$

Aangesien die gekombineerde lengte (kl) in die laaste geval gelyk is aan wl, word die woord korrek as *woorde-boek* verdeel.

Wanneer meer as een geldige kombinasie voorkom, word die verskillende weergawes gekombineer. Byvoorbeeld, vir die woord *aasvoëlnes* met $wl = 10$ word die volgende begin- en eindwoorde onttrek:

BW	EW
aas	nes
aasvoël	voëlnes

Die kombinasies wat hieruit volg is die volgende:

Kombinasie	Gekombineerde lengte
aas-nes	$kl = 6 < wl$
aas-voëlnes	$kl = 10 = wl$
aasvoël-nes	$kl = 10 = wl$
aasvoël-voëlnes	$kl = 14 > wl$

Hier het ons dus twee geldige kombinasies, naamlik *aas-voëlnes* en *aasvoël-nes*. Deur die resultate te kombineer, word die woord ten volle ontbind as *aas-voël-nes*.

3.2.2 Uitbreiding van algoritme

Die basiese prosedure is op 'n woordelys toegepas en die resultate is ontleed. Sekere probleemareas, wat vervolgens bespreek word, is geïdentifiseer en oplossings is daarvoor gevind. Hierdie proses het daartoe gelei het dat die algoritme uitgebrei en verfyn is.

Enkelvoudige woorde verdeel

Die teenwoordigheid van sekere kort woorde in VL, soos *aal*, *bed*, *wens* het veroorsaak dat enkelvoudige woorde verdeel is. Voorbeelde word in Tabel 3.2 getoon.

Hierdie probleem is aangespreek deur die kort woorde wat sulke ongewensde verdelings tot gevolg het uit VL te verwyder. Die woorde is telkens geëvalueer en, na aanleiding van hul

Enkelvoudige woord	Verdeel
aalwyn	aal-wyn
bekoor	bek-oor
benewens	bene-wens
bedarend	bed-arend
kamertjie	kam-ertjie

Tabel 3.2: Enkelvoudige woorde verdeel

invloed op ander woorde in die leksikon tot die begin of einde van woorde beperk. Hiervoor is twee nuwe woordlyste geskep, naamlik een vir woorde wat tot die begin van woorde beperk word (BL) en een vir woorde wat tot die einde van woorde beperk word (EL).

Die woord *bed* veroorsaak byvoorbeeld dikwels verdelingsfoute aan die begin van woorde (*bed-agsaam*, *bed-erflik*, *bed-raad*, ens.). As dit uit VL verwyder word, sal omtrent 60 verdelings in die leksikon gemis word, maar foute soos hierdie word uitgeskakel. Aan die einde van woorde veroorsaak *bed* geen foute nie en ons plaas dit in EL.

Die woord *ertjie* darenteen veroorsaak probleme aan die einde van woorde (*kam-ertjie*, *hang-ertjie*, ens.), terwyl dit nie probleme aan die begin van woorde veroorsaak nie. Slegs 14 verdelings in die leksikon word gemis as dit uit VL verwyder word en in BL geplaas word.

Aangesien die betrokke kort woorde uit VL verwyder is, word die foute nie in die basiese verdelingsalgoritme gemaak nie. Geen saamgestelde woorde wat dit bevat, word egter verdeel nie. In 'n finale stap, wanneer 'n woord sonder hierdie voorde verdeel is, word die begin- en eindlyste (BL en EL) in aanmerking geneem om die samestellende dele wat die kort woorde bevat, verder te verdeel.

Koppeltekens weerskante van konsonante

Woorde met koppeltekens aan weerskante van konsonante het gereeld voorgekom, byvoorbeeld *seun-s-kool*, *tee-r-oos*, *toe-r-oes*, *verdikking-s-laag*, ens. Hierdie verskynsel is die gevolg van twee verskillende geldige verdelings wat gekombineer word.

Byvoorbeeld, vir die woord *toeroes* wat as *toe-roes* verdeel moet word, is die volgende begin- en eindwoorde uit VL onttrek:

SW	EW
toe	oes
toer	roes

Beide kombinasies *toe-roes* en *toer-oes* is geldige verdelings, aangesien $kl = 9 = wl$ in beide gevalle. Wanneer hierdie verdelings gekombineer word, is die resultaat *toe-r-oes*.

Hierdie probleem is op dieselfde manier as die voorafgaande geval hanteer. Die betrokke woorde

is telkens geëvalueer en na aanleiding van hul invloed op ander woorde in die leksikon uit VL verwyder en, waar dit sin maak, in BL of EL geplaas.

Koppeltekens weerskante van kort woordgedeeltes

Kort gedeeltes van woorde het teen die verwagting in tussen koppeltekens voorgekom, soos byvoorbeeld *ys-ter-myn*, *bed-ag-saam*, ens.

Dit gebeur om dieselfde rede as hierbo, byvoorbeeld vir *ystermyn* word *ys-termyn* en *yster-myn* as geldige verdelings verkry. Wanneer dit gekombineer word, is die resultaat *ys-ter-myn*.

Dit is meestal nie moontlik om hierdie probleem uit te skakel nie, maar aangesien dit selde foutiewe woordafbreking veroorsaak, beskou ons dit nie as 'n groot probleem nie en laat dit so. Waar woordafbrekingsfoute as gevolg hiervan mag voorkom, soos in *bed-ag-saam* hierbo, is die woorde in 'n uitsonderingslys geplaas.

Saamgestelde woorde onverdeeld gelaat

Verskillende redes vir saamgestelde woorde wat nie verdeel is nie, is geïdentifiseer. Elkeen van hierdie redes en die manier waarop ons dit aangespreek het, word vervolgens bespreek.

- ▷ **Verbindingsmorfeme**
  In Afrikaans word verbindingsmorfeme dikwels tydens die samestellingsproses tussen woorde ingevoeg. In sulke gevalle kan die gekombineerde lengte van begin- en eindwoorde dus nie met die oorspronklike woordlengte ooreenstem nie en die woord word nie verdeel nie. Byvoorbeeld, vir die woord *stadslewe* met $wl = 9$ word die woorde *stad* en *lewe* uit VL onttrek met gekombineerde lengte $kl = 8$. Dit is dus nodig om voorsiening te maak vir die verbindingsmorfeem *s*.

  Tabel 3.3 toon die verskillende verbindingsmorfeme wat in die algoritme ingesluit is. Ons het dié wat baie selde voorkom en probleme met die toepassing van die algoritme veroorsaak, soos *n*, uitgelaat.

Aantal of letters	Verbindings-morfeem	Voorbeeld
1	s	gebied(s)waters
	e	leeu(e)moed
2	ns	ete(ns)tafel
	er	geselsend(er)wys
3	ens	vertrou(ens)breuk

Tabel 3.3: Verbindingsmorfeme

Die volgende reëls is by die algoritme ingesluit om vir verbindingsmorfeme voorsiening te maak:

Indien

(i) $kl = wl - 1$ en die addisionele letter is *s* of *e*,
(ii) $kl = wl - 2$ en die addisionele letters is *ns* of *er*,
(iii) $kl = wl - 3$ en die addisionele letters is *ens*,

word die woord ná die verbindingsmorfeem verdeel.

▷ **Verbuigings**
Aangesien ons leksikon uit elektroniese korpora saamgestel is, bevat dit nie alle verbuigings[1] van woorde nie. Wanneer 'n saamgestelde woord dus 'n verbuiging bevat wat nie in VL teenwoordig nie, kan dit nie verdeel word nie.

Die woord *spieraanhegtinge* word byvoorbeeld nie verdeel nie aangesien die woord *aanhegtinge* nie in VL voorkom nie. Dit bevat egter die woord *aanheg* wat verbuig is deur die agtervoegsel *tinge*. Hierdie probleem is aangespreek deur die agtervoegsel tydelik te verwyder en die res van die woord as *spier-aanheg* te verdeel. Wanneer die agtervoegsel teruggeplaas word, is die resultaat *spier-aanhegtinge*.

Die verlede tyd van werkwoorde word gewoonlik aangedui deur die voorvoegsel *ge* te gebruik. Die verlede tyd van *bladlees* is byvoorbeeld *gebladlees*. Aangesien die woord *geblad* nie in VL voorkom nie, word die woord nie verdeel nie. Ook hier word die voorvoegsel *ge* tydelik verwyder om *bladlees* te kry wat dan as *blad-lees* verdeel word. Wanneer die voorvoegsel teruggeplaas word, is die resultaat *geblad-lees*.

Lyste van moontlike voorvoegstels (VVL) en agtervoegsels (AVL) is saamgestel om hierdie probleem aan te spreek.

Let wel, die spelling van woorde verander dikwels met verbuiging, byvoorbeeld *onderstreep* wat verbuig word as *onderstreping*. Indien hierdie verbuiging nie so in VL voorkom nie kan 'n woord soos *fraseonderstreping* dus nie verdeel word nie. Alhoewel sulke gevalle gereeld voorkom, veroorsaak dit nie verkeerde woordafbreking nie en ons laat dit vir verdere navorsing.

▷ **Tegnostamme**
Woorde soos *geomagneties*, *elektrokardiogram*, *fisioterapie*, ens. is nie verdeel nie aangesien VL nie die tegnostamme (*geo, elektro, fisio*) woorde bevat nie. Aangesien dit egter betekenis dra, het ons besluit om tegnostamme van die res van die woord te skei indien dit met 'n woord verbind is wat in VL voorkom, soos *hidro-geologie*, *hetero-seksueel*, *tegnopolitieke*, ens. Wanneer dit egter met pseudowoorde soos in *skoop* in *teleskoop*, *logie* in *biologie*, *fiet* in *xerofiet* verbind is, word dit dus nie verdeel nie.

Aangesien tegnostamme altyd aan die begin van woorde voorkom, is die ±500 geïdentifiseerde tegnostamme in BL geplaas.

[1] In Afrikaans word meervoude, verkleining, trappe van vergelyking, verlede tyd, ens. hoofsaaklik deur voor- en agtervoegsels aangedui.

▷ **Komplekse saamgestelde woorde**
Woorde wat uit drie of meer woorde saamgestel is, is dikwels onverdeeld gelaat aangesien die langste woorde wat van die begin en einde van die woord af uit VL onttrek word, die woord nie ten volle dek nie. Beskou byvoorbeeld die woord *skaaphondhanteringskompetisie*. Die langste woord in VL wat met 'n woorddeel van die begin af ooreenstem, is *skaaphond* en van die einde af is dit *kompetisie*. Die gekombineerde lengte van hierdie woorde is te kort en die woord word nie verdeel nie.

Om hierdie probleem aan te spreek, word die langste geïdentifiseerde woord van die begin af (*skaaphond*) tydelik van die res van die woord verwyder en die res (*hanteringskompetisie*) word verdeel. Wanneer die res suksesvol verdeel is (*hanterings-kompetisie*) word die verwyderde gedeelte teruggeplaas en 'n koppelteken word ingeplaas (*skaaphond-hanterings-kompetisie*).

As die res nie verdeel word nie, word dieselfde proses herhaal met die langste geïdentifiseerde woord van die einde van die woord af.

Indien nie een van bogenoemde strategieë 'n verdeling lewer nie, word beide die langste begin- en eindwoorde verwyder en die res word verdeel. Byvoorbeeld, die langste woorde vir die woord *waterkultuurvoedingsmengseltablette* wat in VL voorkom, is *waterkultuur* en *tablette*. Geen verdeling is moontlik wanneer hierdie gedeeltes afsonderlik verwyder word nie. Wanneer beide egter verwyder word, bly *voedingsmengsel* oor wat wel verdeel word as *voedings-mengsel*. Wanneer die verwyderde dele teruggeplaas word, is die resultaat *waterkultuur-voedings-mengsel-tablette*.

Beide *skaaphond-hanterings-kompetisie* en *waterkultuur-voedings-mengsel-tablette* is nie volledig verdeel nie, aangesien *skaaphond* en *waterkultuur* steeds saamgestelde woord is. Die woord word dus deur 'n lus gestuur waar elke deelwoord verdeel word totdat geen verdere verandering plaasvind nie. Die finale resultate vir die twee woorde is *skaap-hond-hanterings-kompetisie* en *water-kultuur-voedings-mengsel-tablette*.

▷ **Verwyderde kort woorde**
Kort woorde wat probleme veroorsaak, is uit VL verwyder en in BL of EL geplaas na aanleiding van hul invloed op ander woorde (Paragraaf 3.2.2). Wanneer hierdie woorde dus in samestellings voorkom, kan dit nie verdeel word nie.

Die woorde in BL en EL mag net aan die begin of einde van woorde voorkom en word as 'n finale stap in die algoritme aangewend. 'n Woord soos *sterfbedwoorde* word byvoorbeeld as *sterfbed-woorde* verdeel aangesien beide *sterfbed* en *woorde* in VL voorkom. Vir die deelwoord *sterfbed* word *sterf* in VL geïdentifiseer en *bed* in EL en die resultaat is die volledig verdeelde woord *sterf-bed-woorde*.

3.3 Die algoritme

Die algoritme is in Perl geprogrammeer (Program D.2). Dit bestaan uit 'n hoofprogram wat vier subroetines vir die verdelingsproses roep. Besonderhede oor die algoritme word in die volgende afdelings bespreek.

3.3.1 Data

Die data vir die verdelingsalgoritme bestaan uit

- ▷ die verwysingslys (VL) waaruit kort woorde en name wat probleme veroorsaak, verwyder is (±181 000 woorde),
- ▷ lyste van woorde wat tot die begin of einde van woorde beperk word, naamlik BL (±500 woorde) en EL (±400 woorde), en
- ▷ lyste van voor- en agtervoegsels, naamlik VVL (±30) en AVL (±170).

3.3.2 Hoofprogram

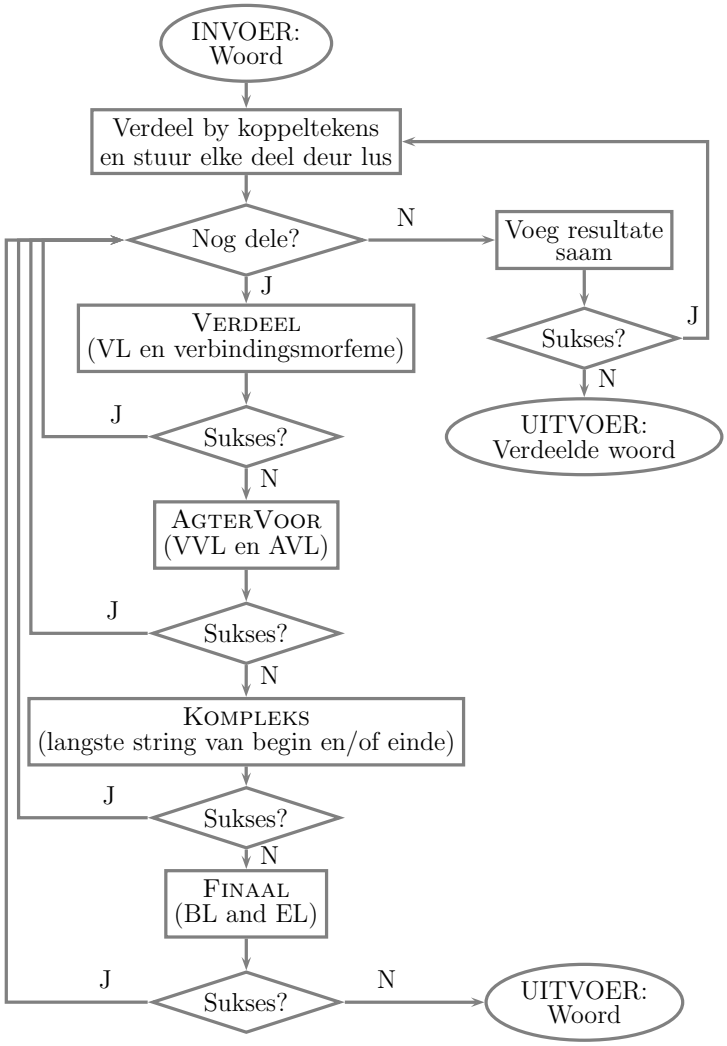
Die vier subroetines wat gebruik word is die volgende:

(i) VERDEEL is die basiese verdeling wat VL en verbindingsmorfeme gebruik.
(ii) AGTERVOOR verdeel woorde deur voor- en agtervoegsels in aanmerking te neem. Dit gebruik VVL en AVL saam met VL.
(iii) KOMPLEKS verdeel komplekse woorde deur langste gemeenskaplike stringe van die begin en/of einde te verwyder en res te verdeel.
(iv) FINAAL verdeel woorde met verwyderde kort woorde in aanmerking geneem. Dit gebruik BL en EL saam met VL.

Figuur 3.1 toon die vloeidiagram van die hoofprogram. Let daarop dat die VERDEEL subroetine telkens geroep word wanneer verdeling in die AGTERVOOR, KOMPLEKS en FINAAL subroetines gedoen word. Verder beteken "sukses" dat 'n verdeling wel plaasgevind het.

'n Woord word by koppeltekens verdeel (indien teenwoordig) en elke deelwoord word deur die verdelingslus gestuur: (i) Begin met basiese verdeling (VERDEEL); (ii) As geen verdeling plaasvind nie, oorweeg voor en agtervoegsels (AGTERVOOR); (iii) As geen verdeling plaasvind nie, verwyder langste gemeenskaplike stringe van begin en/of einde en verdeel die res (KOMPLEKS). (iv) As geen verdeling plaasvind nie, oorweeg kort woorde wat tot begin en einde van woorde beperk word (FINAAL).

Indien die woord deur al die subroetines gegaan het sonder om verdeel te word, word dit as 'n enkelvoudige, onverdeelbare woord beskou en die uitvoer is die oorspronklike woord. Indien dit egter in een van die subroetines verdeel is, word elke deelwoord deur die lus gestuur totdat geen verdere verdeling plaasvind nie. Die uitvoer is dan die volledig verdeelde woord (volgens die algoritme).



Figuur 3.1: Vloeidiagram van die hoofprogram

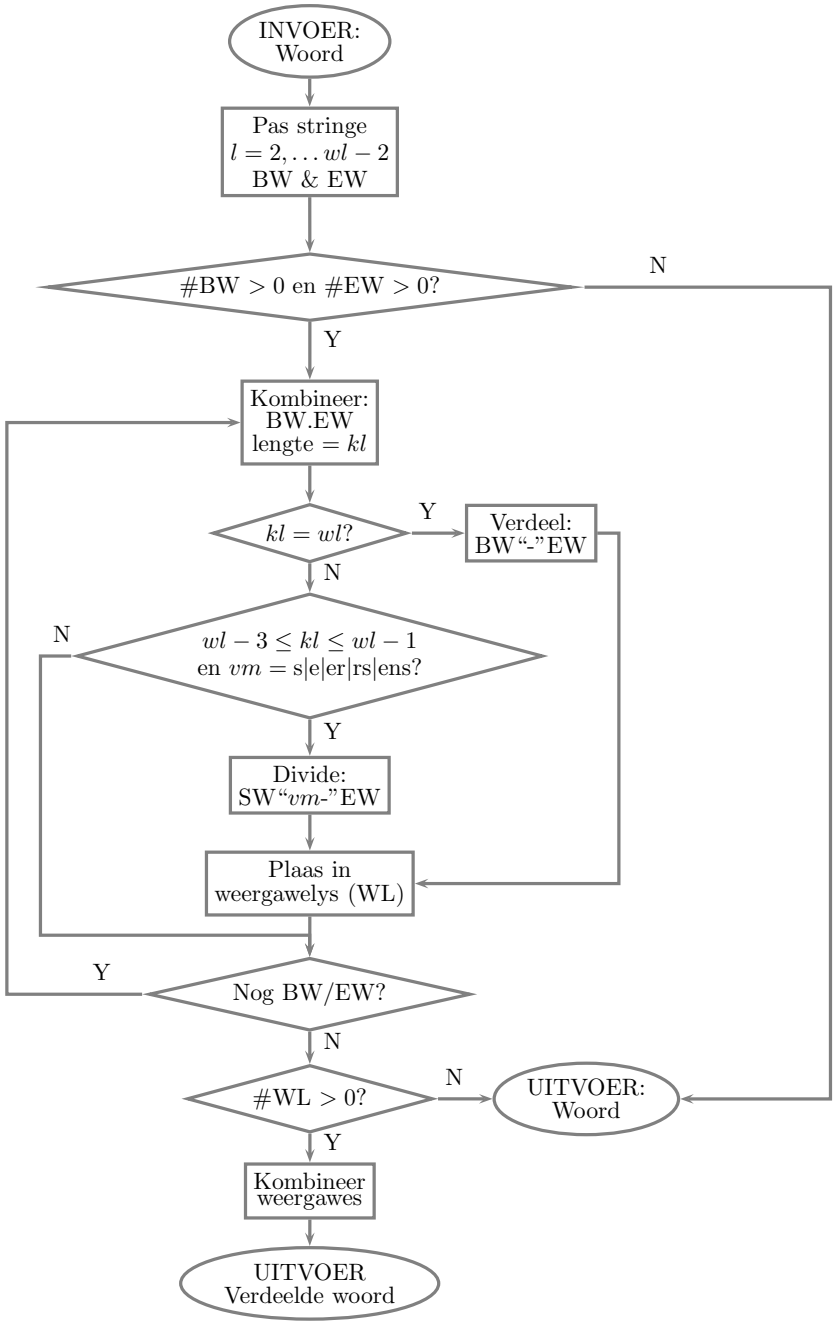
3.3.3 Subroetines

VERDEEL

Figuur 3.2 toon die vloeidiagram van die basiese verdelingsubroetine. Stringe van lengte 2, 3, ..., $wl - 2$ word van die begin en einde van die woord met VL vergelyk en ooreenstemmende deelwoorde word in BW en EW geplaas. As BW en/of EW leeg is, is die uitvoer die oorspronklike woord. As begin- en eindwoorde gevind word (#BW > 0 en #EW > 0) word hulle onderling verbind.

- ▷ Indien $kl = wl$ word die woord verdeel as BW-EW en in die weergawelys (WL) geplaas.
- ▷ As $wl - 3 \leq kl \leq wl - 1$ en die addisionele letters stem met 'n verbindingsmorfeem (vm) ooreen, word die woord ná vm verdeel, dit is BW.vm-EW, en in die weergawelys geplaas.

As alle kombinasies gedoen is, word die geldige weergawes gekombineer en die uitvoer is die verdeelde woord. Indien geen verdeling plaasgevind het nie ($\#WL = 0$) is die uitvoer die oorspronklike woord.

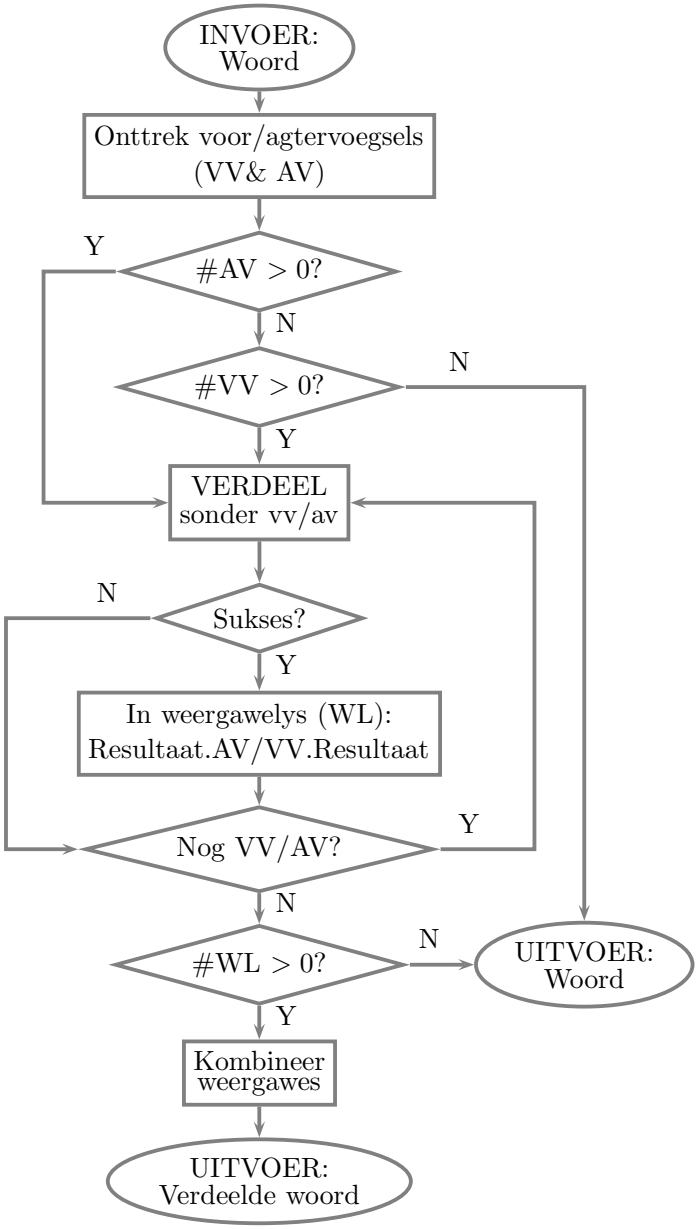


Figuur 3.2: Die subroetine VERDEEL

AGTERVOOR

Figuur 3.3 toon die vloeidiagram van die subroetine AGTERVOOR waar voor- en agtervoegsels in aanmerking geneem word.

Die woord word met VVL en AVL vergelyk en ooreenstemmende voor- en agtervoegsels word in VV en AV geplaas. Indien 'n voorvoegsel (meer as een is onwaarskynlik) en/of agtervoegsels teenwoordig is (#VV > 0 en/of #AV > 0), word dit een-vir-een tydelik verwyder en die res word verdeel. Indien die woord verdeel is, word die voor- of agtervoegsel teruggeplaas en die



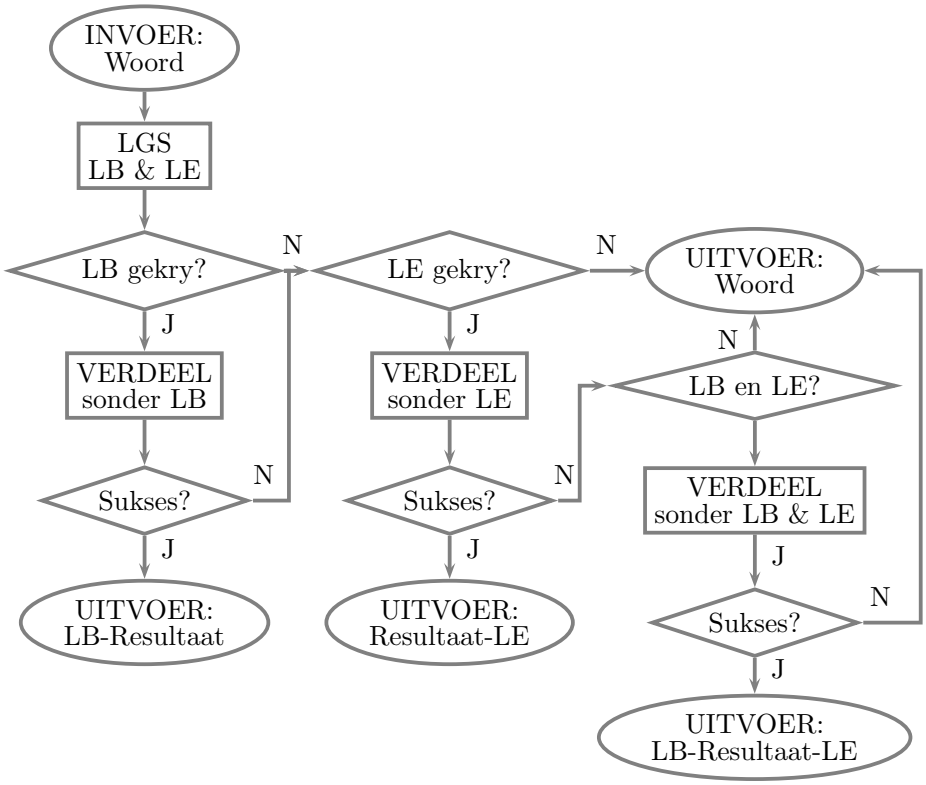
Figuur 3.3: Die subroetine AgterVoor

verdeelde woord word telkens in WL geplaas.

As WL meer as een weergawe bevat, word dit gekombineer en die resultaat word as uitvoer gegee. Indien geen voor- of agtervoegsels teenwoordig is nie, is die uitvoer die oorspronklike woord.

Kompleks

Figuur 3.4 toon die vloeidiagram van subroetine Kompleks.



Figuur 3.4: Die subroetine Kompleks

Deelwoorde word met VL vergelyk en die langste gemeenskaplike stringe van die begin en einde van die woord af (LB en LE) word bepaal. As LB bestaan (#LB> 0), word dit tydelik verwyder en die res word verdeel. As die verdeling suksesvol is, word LB teruggeplaas en met 'n koppelteken geskei.

Indien LB nie bestaan nie en LE wel, word LE tydelik verwyder en die res verdeel. As die verdeling suksesvol is, word LE teruggeplaas en met 'n koppelteken geskei.

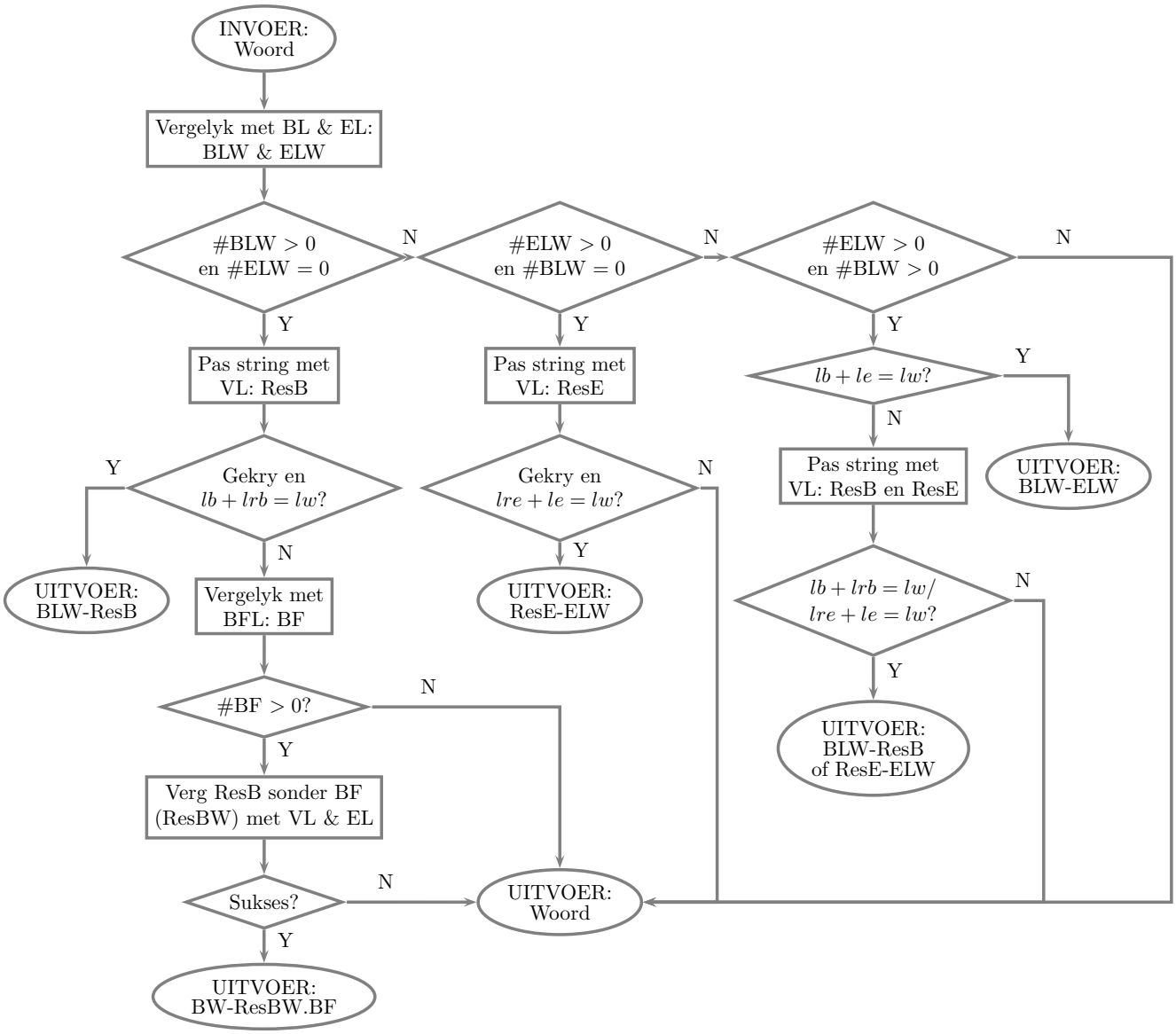
Indien beide LB en LE bestaan, maar geen resultaat is moontlik as dit een-vir-een verwyder word nie, word beide tydelik verwyder en die res word verdeel. As die verdeling suksesvol is, word LB en LE teruggeplaas en met koppeltekens van die resultaat geskei.

As nóg LB, nóg LE bestaan, of as geen suksesvolle verdeling plaasvind nie, word die oorspronklike woord as uitvoer gegee.

Finaal

Figuur 3.5 toon die vloeidiagram van die Finaal subroetine.

Agtervoegsels word by woorde wat in EL voorkom in ag geneem, maar geen voor- of agtervoegsels word in aanmerking geneem by woorde uit BL nie. Verder word verbindingsmorfeme nie in die finale stap in aanmerking geneem nie sodat foute soos *kerk-s-keuring* uitgeskakel word.



Figuur 3.5: Die subroetine FINAAL

Die woord word met BL en EL vergelyk om woorde wat tot die begin of einde van woorde beperk word, te identifiseer en onderskeidelik in BLW en ELW te plaas.

As 'n beginwoord gevind is (#BLW > 0) maar geen ELW nie, word BL verwyder en die res van die woord (ResB) word met VL vergelyk. Indien dit in VL voorkom en die lengte van BLW en ResB saam is gelyk aan die woordlengte ($lb + lrb = wl$) word die woord verdeel as BLW-ResB. As ResB nie in VL is nie, word bepaal of dit 'n agtervoegsel bevat deur dit met AVL te vergelyk. As #AV$>$ 0 word die agtervoegsel verwyder en ResB sonder die agtervoegsel (ResBAV) word

met EW en VL vergelyk. Indien suksesvol, word die woord verdeel as BW-ResBAV.AV. Hier dui die "." daarop dat ResBAV en AV aanmekaargeskryf word.

As #ELW > 0 en #BLW$= 0$, word die woord sonder ELW (ResE) met VL vergelyk. Indien 'n ooreenstemmende woord gevind word en die lengte van ELW (le) plus die lengte van ResE (lre) is gelyk aan die woordlengte ($lre + le = wl$), word die woord verdeel as ResE-ELW. Indien nie, word die oorspronklike woord as uitvoer gegee.

As beide #BLW > 0 en #ELW > 0 en $lb + le = lw$, is die uitvoer BLW-ELW. Indien nie, word ResB en ResE met VL vergelyk. As $lb + lrb = lw$ of $lre + le = lw$ is die uitvoer BLW-RestB of RestE-ELW, respektiewelik.

Let daarop dat meer as een woord uit die beginlys (BLW), die eindlys (ELW) of meer as een agtervoegsel (AV) mag voorkom. Die lusse om deur hierdie moontlikhede te itereer word nie in die vloeidiagram getoon nie.

3.4 Prestasiemaatstawwe

Vir beide die saamgesteldewoord- en lettergreepverdelingstake beskou ons prestasie ten opsigte van volledige woorde en verdelingsgeleenthede (elke posisie tussen letters) as belangrik. Die prestasiemaatstawwe wat gebruik word, is gebaseer op prestasiemaatstawwe wat in inligtingsherwinning gebruik word [MRS08] en word in terme van die volgende uitkomste bepaal:

- ▷ Woorde reg verdeel (koppeltekens reg ingeplaas) – Reg positief (C_p)[2];
- ▷ Woorde reg onverdeeld gelaat (koppeltekens reg weggelaat) – Reg negatief (C_n);
- ▷ Woorde met foutiewe verdelings (koppeltekens foutief ingeplaas) – Fout positief (F_p);
- ▷ Woorde met verdelings gemis (koppeltekens gemis) – Fout negatief (F_n).

Die prestasiemaatstawwe word soos volg gedefinieer.

(a) Akkuraatheid – die waarskynlikheid dat verdeelbare woorde reg in lettergrepe verdeel word en dat onverdeelbare woorde onverdeeld gelaat word (of dat koppeltekens reg ingeplaas of weggelaat word), met

$$A = \frac{C_p + C_n}{C_p + C_n + F_p + F_n}.$$

Dit is dus alle gevalle van korrekte gedrag gedeel deur alle moontlike gevalle.

(b) Presisie – die waarskynlikheid dat verdeelbare woorde reg verdeel word (of dat koppeltekens reg ingeplaas word), met

$$P = \frac{C_p}{C_p + F_p}.$$

Dus alle gevalle waar koppeltekens reg ingeplaas is, gedeel deur alle gevalle waar koppeltekens ingeplaas is – reg en verkeerd.

[2]Ons behou die Engelse afkortings ter wille van kruisverwysing met bestaande literatuur.

(c) Herroeping (*Recall*) – die waarskynlikheid dat verdeelbare woorde reg verdeel word en nie onverdeeld gelaat word nie (of dat koppeltekens reg ingeplaas word en nie weggelaat word nie), met

$$R = \frac{C_p}{C_p + F_n}.$$

Dus alle gevalle waar koppeltekens reg ingeplaas is, gedeel deur die gevalle waar koppeltekens reg ingeplaas en gemis is.

(d) F-telling – die tradisionele, of gebalanseerde, geweegde harmoniese gemiddelde van presisie en herroeping, met

$$F = 2 \times \frac{P \times R}{P + R}.$$

3.5 Resultate

Om data te bekom waarteen die algoritme getoets kon word, is die saamgestelde woorde in die leksikon van ±183 000 woorde per hand verdeel en gekontroleer. Hierdie lys word as grootliks korrek beskou. Ons het die algoritme op die volledige lys toegepas. Dit het 12 uur, 15 minute en 32 sekondes op die *High Performance Computer* (HPC) by Unisa geneem – gemiddeld 0,24 sekondes per woord. Statistiek met betrekking tot volledige woorde sowel as verdelingsgeleenthede[3] is bekom.

Die resultate ten opsigte van volledige woorde word in Tabel 3.4 getoon.

		Volledige woorde	
		Saamgesteld	Enkelvoudig
Algoritme	Verdeel	C_p 89 620 (84%)	F_p 87 (0,1%)
	Verdeel nie	F_n 17 619 (16%)	C_n 75 107 (99,9%)
Totaal		107 239 (59%)	75 194 (41%)

Tabel 3.4: Resultate ten opsigte van volledige woorde

Die resultate ten opsigte van verdelingsgeleenthede word in Tabel 3.5 getoon.

Die prestasie van die algoritme met betrekking tot volledige woorde en verdelingsgeleenthede word in Tabel 3.6 getoon.

[3]Die hooflys bevat ±2 000 000 posisies tussen letters wat as verdelingsgeleenthede beskou word.

		Verdelingsgeleenthede	
		Koppelteken	Geen koppelteken
Algoritme	Koppelteken	C_p 100 049 (84%)	F_p 87 (0,005%)
	Geen koppelteken	F_n 18 785 (16%)	C_n 1 913 491 (99,995%)
Totaal		118 834	1 913 578

Tabel 3.5: Resultate ten opsigte van verdelingsgeleenthede

Maatstaf	Volledige woorde	Verdelings-geleenthede
Akkuraatheid	90,3%	99,1%
Presisie	99,9%	99,9%
Herroeping	83,6%	84,2%
F-telling	91,0%	91,4%

Tabel 3.6: Prestasie van die algoritme

Dis duidelik dat die algoritme goed presteer. Presisie op volledige woorde en verdelingsgeleenthede is 99,9% wat beteken dat die kans dat die algoritme 'n saamgestelde woord verkeerd sal verdeel (of 'n koppelteken op 'n verkeerde plek sal plaas) 0,1% is.

Akkuraatheid op volledige woorde is 90,3% wat beteken dat die waarskynlikheid dat 'n woord verkeerd verdeel sal word minder as 10% is. Akkuraatheid op verdelingsgeleenthede is 99,1% wat beteken dat daar 'n waarskynlikheid van minder as 1% is dat 'n posisie tussen letters verkeerd geklassifiseer sal word.

Die relatief lae herroeping is die gevolg van saamgestelde woorde wat onverdeeld gelaat word. Die hoofredes hiervoor is (i) die kort woorde wat uit VL verwyder is, en (ii) verbuigings met veranderde spelling wat nie in VL voorkom nie.

3.5.1 Bespreking

Die prestasie van die algoritme word vervolgens met voorbeelde geïllustreer. Hier word 'n koppelteken wat verkeerd ingeplaas is deur 'n asterisk (*) aangedui en 'n koppelteken wat gemis is deur 'n uitroepteken (!).

Verdelingsfoute sonder negatiewe gevolge

Uit die ±183 000 woorde is daar slegs 87 met koppeltekens wat verkeerd ingeplaas is, waarvan slegs 16 gevalle tot 'n verkeerde afbreking sal lei. Die res (71 woorde) bevat foutiewe koppeltekens wat met geldige woordafbrekingspunte ooreenstem.

- ▷ Die volgende is enkelvoudige woorde, maar is verdeel omdat woorddele dieselfde is as woorde in VL, BL of EL.

lid*maat	pas*poort	spek*takels
mos*kou	span*spek	toer*nooi

- ▷ In die volgende woorde word tegnostamme (*geo* en *hiper*) van pseudowoorde wat nie in VL of EL voorkom nie geskei.

  geo*tropisme hiper*bolies

  Verbuigings van die woorde *trop* en *bol* met die agtervoegsels *isme* en *ies*, onderskeidelik, word as woorde beskou en die enkelvoudige woorde word dus verdeel.

- ▷ In die volgende gevalle is twee geldige verdelings gekombineer:

  ys*ter-myn gevoels-in*druk

  Vir *gevoelsindruk*, byvoorbeeld, is beide *gevoels-indruk* en *gevoelsin-druk* geldige verdelings volgens die algoritme. As dit gekombineer word, is die resultaat *gevoels-in-druk*.

- ▷ Die regte verdeling vir *inkpotlood* is *ink-potlood*, maar die algoritme lewer

  ink!pot*lood.

  Die woord *ink* is uit die lyste verwyder en die woorde *inkpot* en *lood* word onttrek sodat die woord as *inkpot-lood* verdeel word.

Verdelingsfoute wat tot afbrekingsfoute mag lei

- ▷ Die volgende woorde is verdeel met koppeltekens aan weerskante van konsonante:

  hawe*r-oes val*k-uil

  Dit is die gevolg van twee geldige weergawes wat gekombineer word. Vir *haweroes* word *hawe-roes* en *hawer-oes* gekombineer en vir *valkuil* *val-kuil* en *valk-uil*. Hier berus die korrekte verdeling op die konteks waarin die woord gebruik word en die gebruiker moet besluit watter een om te gebruik.

- ▷ Die volgende foute is as gevolg van dieselfde rede as die eerste punt hierbo. Woorddele wat dieselfde is as woorde in VL, BL of EL kom in die woorde voor.

nag*raads	petrol*oog
toer*oes	volk*s!telling

  Ons sou hierdie foute kon uitskakel deur die betrokke woorde (*nag*, *oes*, *oog*, *stelling*) te verwyder, maar dit sou baie gemiste verdelings tot gevolg hê. Aangesien daar so min

foute van hierdie aard is, kan dit as uitsonderings beskou word en in 'n uitsonderingslys geplaas word.

Gemiste verdelings

Baie verdelings is deur die algoritme gemis (in 9,66% van die woorde). Dit is hoofsaaklik die gevolg van woorde wat uit VL verwyder is, verbuigings met veranderde spelling en eiename wat nie in VL voorkom nie.

- ▷ Probleemwoorde is verwyder om verdelingsfoute uit te skakel, met die gevolg dat geen verdeling plaasvind waar hierdie woorde betrokke is nie.

aand!rok	groen!ertjies	hoek!skop
glad!skuur	haai!tand	wonder!kuur

- ▷ Woorde soos *onderwys*, *deurmekaar* en *voorwerp* is enkelvoudige woorde en behoort nie verdeel te word nie. Voorvoegsels soos *onder*, *deur* en *voor* is uit ons lyste verwyder en woorde wat dit bevat, is dus nie verdeel nie.

  onder!afdelings deur!wagter voor!wiel

- ▷ Verbuigings met veranderde spelling wat nie in VL voorkom nie veroorsaak dat saamgestelde woorde wat dit bevat, nie verdeel word nie.

  meer!taligheid veel!fasige drie!delige

- ▷ Ons lyste bevat nie alle eiename en woorde wat selde gebruik word nie. Samestellings met sulke woorde word dus nie verdeel nie.

  konstruk!geldigheids!modelle himenium!laag grimbeek!park

Lang woorde reg verdeel of onverdeeld gelaat

Die volgende is voorbeelde van lang saamgestelde woorde wat die algoritme suksesvol verdeel het:

obligasie-mark-termyn-handels-kontrakte	leer-program-ontwikkelings-proses
meervoudige-probleem-siekte-toestand	video-band-terug-speel-apparaat
hulp-bron-samewerkings-ooreenkoms	geel-bek-neus-horing-voël

Ewe belangrik is voorbeelde van lang enkelvoudige woorde wat reg onverdeeld gelaat is:

kompartementalisering	gedifferensieerdheid
geïnstitusionaliseerde	verdiskonteerbaarheid

In die volgende hoofstukke word die drie masjienleertegnieke, neurale netwerke, beslissingsbome en die TEX-algoritme in detail bespreek. Dit sluit die ontwikkeling van optimale modelle vir beide die lettergreep- en saamgesteldewoordverdeling in. Daarna volg ook 'n vergelyking van die prestasie van die drie tegnieke.

Ons gebruik telkens 'n toetsdatastel van $\pm 10\,000$ woorde wat lukraak uit die lettergreep- en saamgesteldewoordverdeelde leksikons onttrek is vir toetsing en gebruik die oorblywende $\pm 173\,000$ woorde vir afrigting.

Soos vroeër gemeld, volg ons 'n pragmatiese benadering by die evaluering van prestasie tydens die ontwikkeling van die optimale modelle en ook by die vergelyking van die tegnieke. Ons poog dus nie om aan te toon dat een tegniek statisties beduidend beter as 'n ander op die probleem onder beskouing presteer nie.

Hoofstuk 4

Kunsmatige Neurale netwerke

In hierdie hoofstuk bespreek ons kunsmatige neurale netwerke as patroonherkenningstegniek. Ons begin deur agtergrond oor die inspirasie vir en ontwikkeling van sulke neurale netwerke te verskaf. Daarna beskryf ons die metode wat gevolg is om afrigtingsdata vir die lettergreepverdelingstaak te genereer asook die proses wat gevolg is om 'n optimale neurale netwerk vir hierdie taak te ontwikkel. Hierdie optimale neurale netwerk is met $\pm 10\,000$ woorde wat as toetsdata uitgehou is, getoets. Die resultate hiervan word getoon en bespreek. Laastens bespreek ons ook 'n neurale netwerk wat vir saamgesteldewoordverdeling afgerig en getoets is.

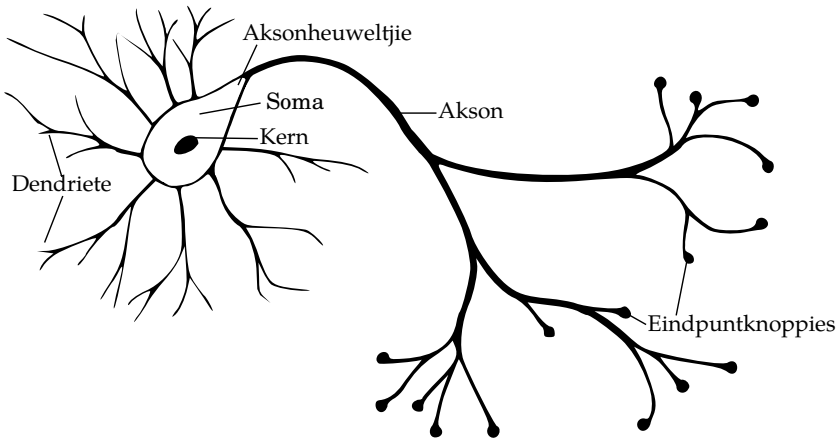
4.1 Agtergrond

In die 1950s het die Amerikaners probeer om rekenaars te gebruik om teks uit vreemde tale, soos Russies, outomaties in Engels te vertaal. Masjienvertaling het egter nie uit die staanspoor akkurate vertalings gelewer nie en dit het duidelik geword dat die outomatiese taalverwerking meer kompleks is as wat aanvanklik aangeneem is. Die veld van rekenaarlinguistiek het gevolglik ontstaan as 'n nuwe studierigting waar algoritmes en programmatuur ontwikkel word om taaldata op 'n intelligente wyse te verwerk. Sedert die ontsaan van die dissipline Kunsmatige Intelligensie (KI) in die 1960s, is rekenaarlinguistiek daardie onderafdeling van KI wat poog om natuurlike taalverwerking te doen soos mense dit sou doen [Wik13a].

Kunsmatige neurale netwerke is een van die KI-tegnieke wat wyd in rekenaarlinguistiek gebruik word. Die werking van biologiese senuweestelsels het as inspirasie gedien vir die ontwikkeling van hierdie tegniek.

4.1.1 Biologiese neurale netwerke

Die menslike brein bevat ongeveer 10 miljard neurone (of senuweeselle) wat onderling met mekaar verbind is. Figuur 4.1 toon 'n vereenvoudigde skematiese voorstelling van 'n biologiese neuron.



Figuur 4.1: Skematiese voorstelling van 'n biologiese neuron [Fra98]

In 'n biologiese neuron ontvang *dendriete* impulse vanaf ander neurone, die *soma* sommeer en verwerk die inligting en wanneer 'n neuron "vuur" word 'n impuls langs die *akson* uitgestuur.

Waar 'n neuron se *eindpuntknoppie* en 'n naburige neuron se dendriet bymekaarkom, is daar 'n gaping wat 'n *sinaps* genoem word. Alle aksies in die brein vind by die sinapse plaas en geheue word in die sinaptiese verbindings in terme van elektriese en chemiese reaksies geënkodeer.

Tydens seinprosessering word seine vanaf ander neurone deur 'n neuron se dendriete versamel en deur die soma bymekaargetel. Beide ruimtelike en temporale sommasie vind plaas. By ruimtelike sommasie word verskeie swak seine in een sterk sein omgeskakel en by temporale sommasie word kort opmekaar volgende swak impulse vanaf dieselfde bron in een sterk sein omgeskakel [Fra98].

Indien genoegsame invoer ontvang word sodat die drempel van die aksonheuweltjie oorskry word, "vuur" die neuron. Neuro-oordraers (chemiese boodskappers) word vrygestel en 'n elektriese sein word langs die akson na ander neurone, of na strukture buite die senuweestelsel (soos spiere) oorgedra. Indien die invoere nie genoeg is om die drempel te oorskry nie, verval dit vinnig en geen sein word gegenereer nie.

Die neuro-oordraers versprei oor die sinapse en verbind met naburige neurone. Elke neuron kan duisende verbindings soos hierdie vorm sodat 'n tipiese brein sowat 100 triljoen sinapse bevat. Die punte van 'n neuron wat na naburige neurone uitreik (dendriete) ontvang elektriese impulse vanaf ander neurone.

Die verbindings tussen neurone vorm 'n netwerk en verander gedurig. Wanneer een neuron 'n sein na 'n ander neuron stuur, word die verbinding (sinaps) tussen die twee sterker. Hoe meer seine tussen hulle gestuur word, hoe sterker word die verbinding.

Soos 'n mens leer en die wêreld ondervind, vind verandering by sinapse en dendriete plaas en nuwe verbindings word gevorm. Die brein is dus in staat om ditself voortdurend te herstruktureer. Hierdie vervormbaarheid (plastisiteit) van die brein stel dit in staat om te herstel indien dit beskadig word.

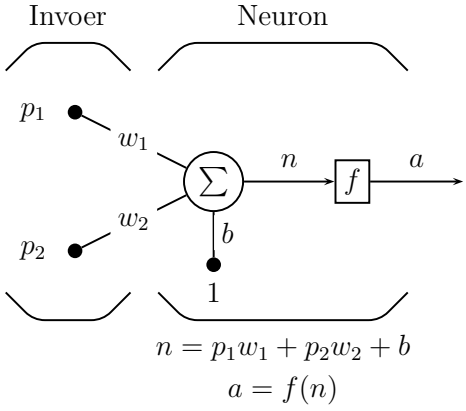
4.1.2 Kunsmatige neurale netwerke

Volgens Haykin [Hay09] is 'n kunsmatige neurale netwerk 'n verwerker wat uit parallelverspreide, eenvoudige verwerkingseenhede bestaan. Dit het die vermoë om kennis wat uit voorbeelde van 'n bepaalde probleem bekom is, te stoor en beskikbaar te stel vir gebruik. Dit kom met twee eienskappe van die brein ooreen, naamlik dat dit kennis bekom deur 'n leerproses en dat hierdie kennis in die sterkte van die verbindings tussen die neurone gestoor word.

Kunsmatige neurale netwerke – voortaan bloot neurale netwerke (NN) genoem – bestaan dus uit *neurone* wat in parallelle lae gerangskik is en seine oor geweegde verbindings (sinapse) ontvang, sommeer, verwerk en aanstuur.

Neurone

Figuur 4.2 toon 'n eenvoudige neurale netwerk wat uit een neuron bestaan. Die neuron ontvang twee invoerelemente, p_1 en p_2, wat die eienskappe, of veranderlikes, van 'n probleem verteenwoordig, oor verbindings met gewigte w_1 en w_2, asook 'n invoer van waarde een oor 'n verbinding met belading b. 'n Belading is soortgelyk aan 'n gewig, maar het 'n konstante invoer van een. Dit verskuif die funksie f met b eenhede sodat dit nie altyd deur die oorsprong gaan nie.



Figuur 4.2: 'n Enkelneuronnetwerk

Die geweegde invoere en belading word deur die neuron gesommeer om die *netto invoer*, n, te kry, met $n = p_1w_1 + p_2w_2 + b$. Die *uitvoer* van die neuron, a, word bereken deur 'n oordragfunksie f op die netto invoer toe te pas ($a = f(n)$). Oordragfunksies word gekies na aanleiding van die probleem onder beskouing. Die streng beperkende trapfunksie lewer byvoorbeeld waardes van nul of een, terwyl die logaritmiese sigmoidale oordragfunksie kontinue waardes tussen nul en een as uitvoer lewer.

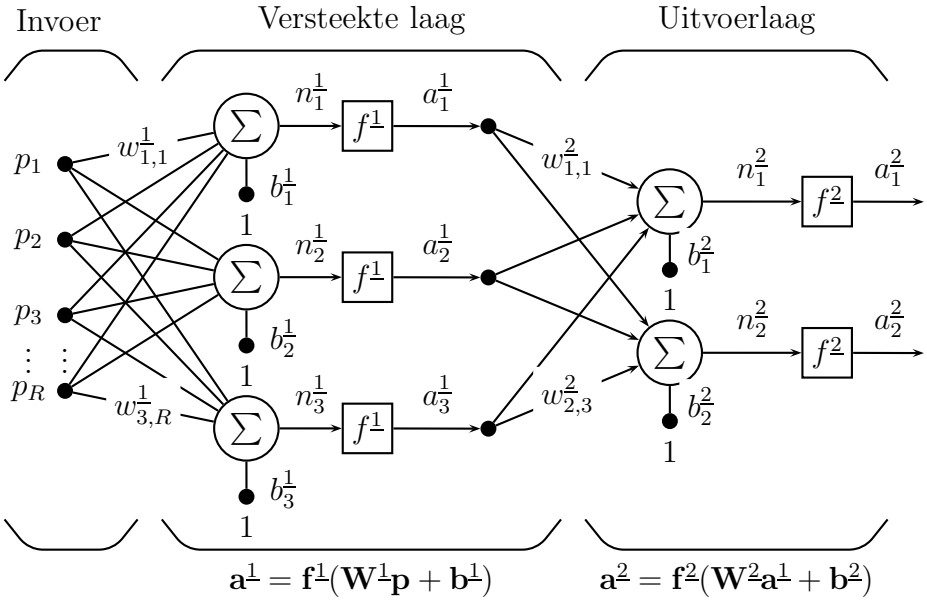
Neuronlae

In 'n neurale netwerk word neurone in lae gerangskik. Die neurone in 'n laag kommunikeer nie met mekaar nie, maar elke neuron verwerk die invoere wat dit ontvang. Indien daar meer as een neuronlaag is, word die uitvoer van een laag as invoer vir 'n volgende laag aangestuur. Die uitvoer van die laaste laag is ook die uitvoer van die netwerk.

Ons onderskei tussen twee soorte neuronlae, naamlik *versteekte* en *uitvoerlae*.

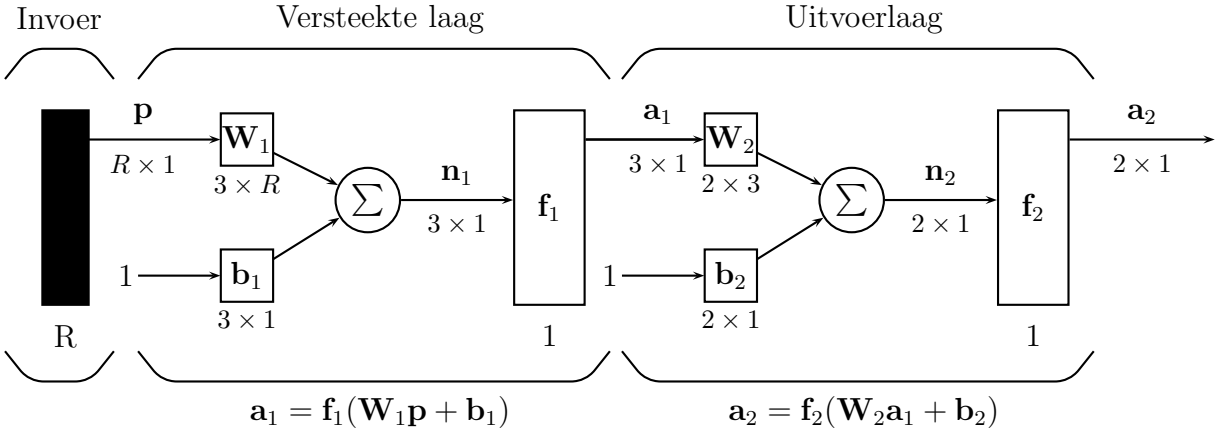
- ▷ Neurone in 'n versteekte laag ontvang invoer van buite die netwerk en in die geval van meerlaagnetwerke vanaf neurone in voorafgaande versteekte neuronlae.
- ▷ Neurone in die uitvoerlaag ontvang invoer vanaf die laaste versteekte laag in die netwerk. Dit verwerk hierdie invoer en lewer die uitvoer van die netwerk.

In Figuur 4.3 word 'n tweelaag-neuralenetwerk met drie neurone in die versteekte laag en twee neurone in die uitvoerlaag getoon. Ons gebruik die notasie 3–2 om so 'n netwerk voor te stel. Die netwerk ontvang R invoerelemente van buite af. In hierdie netwerk word gewigte, netto invoer, oordragfunksies en uitvoere van laag een en twee van mekaar onderskei deur onderstreepte boskrifte te gebruik.



Figuur 4.3: 'n 3–2-NN

In die praktyk bevat neurale netwerke gewoonlik baie – dikwels honderde en selfs duisende – invoerelemente en versteekte neurone, waarvoor die notasie soos in Figuur 4.3 nie bruikbaar is nie. In Figuur 4.4 word hierdie 3–2-neuralenetwerk in bloknotasie getoon, soos dit in Matlab se gebruikersgids gebruik word. Die vektore en matrikse wat invoer, gewigte, beladings, netto invoer, oordragfunksies en uitvoer verteenwoordig, word hier as blokke voorgestel. Aangesien individuele elemente nie meer getoon word nie, is dit ook nie nodig om die boskrifte vir laagidentifikasie te gebruik nie, en word verskillende lae deur voetskrifte aangedui.



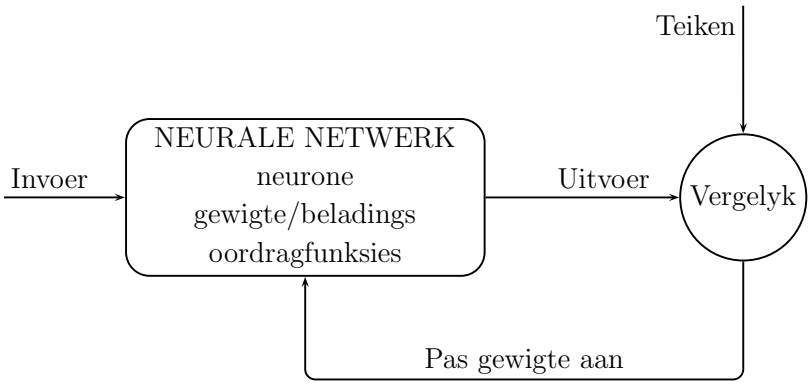
Figuur 4.4: Die 3–2-neuralenetwerk in bloknotasie

4.1.3 Afrigting

'n Mens se geheue word in die sinaptiese verbindings tussen neurone in die brein vasgelê. Ook in neurale netwerke word die funksie wat dit verrig deur die gewigte op die verbindings tussen die neurone van die netwerk bepaal.

Twee soorte afrigting woord onderskei, naamlik afrigting met kontrole en afrigting sonder kontrole[1]. By gekontroleerde afrigting word die gewensde uitvoer vir elke invoerdatapunt verskaf (ons noem dit teikendata). Die netwerk se uitvoer vir 'n sekere invoer word telkens met die ooreenstemmende teikendata vergelyk en gewigte word aangepas totdat die uitvoer genoegsaam met die teiken ooreenstem.

Figuur 4.5 toon 'n skematiese voorstelling van gekontroleerde afrigting.



Figuur 4.5: Gekontroleerde afrigting

[1]Afrigting sonder kontrole val buite die bestek van hierdie studie. Hier word gewigte aangepas in reaksie op die invoere – gewoonlik na aanleiding van patrone wat in die invoerdata voorkom.

4.2 Lettergreepverdeling

Lettergreepverdeling is in wese 'n patroonherkenningstaak. Aangesien dit bekend is dat neurale netwerke by uitstek goed is met patroonherkenning, is dit 'n sinvolle tegniek om vir lettergreepverdeling te oorweeg.

Die moontlikheid om 'n neurale netwerk vir hierdie probleem te gebruik is reeds voorheen ondersoek, met gunstige resultate [Fic02, DvdB92]. In hierdie studie het ons 'n neuralenetwerkmodel wat op 'n volledige karakterstel van 46 karakters vir Afrikaans gebaseer is ontwikkel en soveel moontlik data vir afrigting gebruik.

4.2.1 Afrigtingsdata

Ons leksikon van ±183 000 woorde wat in lettergrepe verdeel is, is gebruik om 'n neuralenetwerkmodel vir lettergreepverdeling te ontwikkel. Hieruit is 'n toetsdatalys van ±10 000 woorde onttrek, terwyl die oorblywende ±173 000 woorde vir afrigting gebruik is. Dit is belangrik dat die data wat vir die ontwikkeling van neurale netwerke gebruik word so skoon moontlik moet wees. Program D.3.1 is gebruik om afrigtingsdata te genereer.

Data vir die afrigting van neurale netwerke is uit woorde in die afrigtingsdatastel gegenereer deur 'n venster van bepaalde lengte oor elke woord te skuif terwyl 'n teikenwaarde vir elke venster onttrek is. In Figuur 4.6 word getoon hoe invoerdata van lengte agt en die geassosieerde teikenwaarde uit die woord *let-ter-gre-pe* onttrek word. Die teikenposisie word deur 'n dubbele vertikale lyn aangedui, met ander woorde daar is vier karakters vóór en vier ná die teikenposisie.

Venster								**Teiken**
			l	e	t	t	e	0
		l	e	t	t	e	r	0
	l	e	t	t	e	r	g	1
l	e	t	t	e	r	g	r	0
e	t	t	e	r	g	r	e	0
t	t	e	r	g	r	e	p	1
t	e	r	g	r	e	p	e	0
e	r	g	r	e	p	e		0
r	g	r	e	p	e			1
g	r	e	p	e				0

Figuur 4.6: Opeenvolgende 4–4-vensters (met teikens) vir die woord *let-ter-gre-pe*

4.2.2 Enkodering

Die gegenereerde vensters is vir die neurale netwerk geënkodeer sodat die invoer uit nulle en ene bestaan. Tabel 4.1 toon die 46 karakters wat in vensters wat uit Afrikaanse woorde gegenereer word, mag voorkom. Dit sluit die volgende in: (i) die 26 letters van die alfabet, (ii) klinkers met diakritiese tekens (17) en (iii) spesiale karakters (3).

1 ␣	2 a	3 á	4 ä	5 e	6 é	7 è	8 ë	9 ê	10 i	11 í	12 ï	13 î	14 o	15 ó	16 ò
17 ö	18 ô	19 u	20 ú	21 ü	22 û	23 y	24 ý	25 b	26 c	27 d	28 f	29 g	30 h	31 j	32 k
33 l	34 m	35 n	36 p	37 q	38 r	39 s	40 t	41 v	42 w	43 x	44 z	45 =	46 ’		

Tabel 4.1: Karakters (46) wat in vensters mag voorkom

Volgens ’n vorige studie [SS96] lewer enkodering waarin die klinkers (ingesluitend alle diakritiese variasies) apart van die medeklinkers saam gegroepeer word, beter resultate. Ons het dus ’n soortgelyke rangskikking gebruik deur die klinkers saam eerste te lys (kleiner waardes). Die drie nie-letterkarakters in die tabel verteenwoordig die volgende: die spasie (␣) wat in vensters ingevoer word om te verseker dat alle tussenletterposisies in aanmerking geneem word, die harde koppelteken (=) wat in woorde soos *luitenant-generaal* voorkom en die afkappingsteken (’) wat in woorde soos *foto’tjie* voorkom.

Die harde koppelteken is nodig om duidelik onderskeid te maak tussen koppeltekens wat lettergreepverdelings aandui – en dus diskresionêr aangewend word as woordafbreking nodig is – en koppeltekens wat nie uit woorde verwyder moet word nie.

Tydens enkodering word elkeen van die karakters in ’n venster deur ’n string van nulle en ene verteenwoordig, met ’n 1 in die posisie waar die letter in die rangskikking voorkom en die res nulle. Byvoorbeeld, die letter *e* wat in posisie 5 voorkom, word as ’n string van 45 nulle en ’n 1 in posisie 5 voorgestel:

0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

’n Geënkodeerde venster van agt karakters bestaan dus uit $8 \times 46 = 368$ karakters waarvan slegs agt ene is en die res (360) nulle. Die enkodering vir die tweede venster in Figuur 4.6 (␣ ␣ l e t t e r) is dus soos volg:

1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

4.2.3 Afrigting

Wanneer 'n neurale netwerk afgerig word, word geënkodeerde vensters in die neurale netwerk ingevoer. Die neurale netwerk gebruik die huidige gewigte en beladings wat aanvanklik ewekansig gegenereer word en 'n oordragfunksie om vir elke invoer 'n uitvoer te bepaal. Hierdie uitvoer word telkens met die ooreenstemmende teikenwaarde vergelyk en die gewigte word aangepas. Hierdie proses word herhaal totdat die uitvoer genoegsaam met die teiken ooreenstem. Sien die skematiese voorstelling op bladsy 49.

4.2.4 Matlab

Matlab se *Neural Networks Toolbox* [Mat13] is vir die afrigting van neurale netwerke gebruik. In hierdie proefskrif gebruik ons ook dieselfde notasie as in Matlab se gebruikersgids om neurale netwerke te beskryf, waar die invoer nie as 'n laag in die netwerk beskou word nie. 'n Neurale netwerk met een versteekte laag is dus 'n tweelaagnetwerk – die versteekte en die uitvoerlaag.

Die volgende is 'n voorbeeld van 'n Matlab-program wat vir die afrigting van 'n neurale netwerk vir die lettergreepverdelingsprobleem gebruik is:

```
P = Invoer'; T = Teiken';                          %Invoerdata
rng('shuffle');                                    %Verander "saad" vir kansgetalle
net = newff(P,T,[80],{'logsig', 'logsig'},
          'trainrp','learngdm','mse',{},{});
net.trainParam.goal = 0.01;                        %Doelwit: MSE<0,01.
net.trainParam.epochs = 1000;                      %Maksimum epogge
net.trainParam.min_grad = 1e-10                    %Minimum toegelate gradient
net.divideFcn='';                                  %Verdeling vir validasie en toetsing
[net,tr] = train(net,P,T);                         %Rig af
save netNEUR net;                                  %Stoor afgerigte NN
```

Let op die volgende:

- ▷ Die invoer word deur Perl in rye gegenereer, terwyl Matlab dit in kolomme inlees. Die getransponeerde van die invoerdatalêers word dus gebruik (`Invoer'` en `Teiken'`).
- ▷ Sedert 2011 bly die saad van die kansgetalgenerator in Matlab dieselfde tensy dit programmaties verander word. Ten einde nie elke keer dieselfde neurale netwerk te kry wanneer 'n netwerk met 'n spesifieke konfigurasie afgerig word nie, moet die saad verander word.
- ▷ Die neurale netwerk wat geskep word ('n vorentoevoer terugverspreidingsnetwerk, aangedui deur `newff`) gebruik P en T as invoerdata; dit bevat een versteekte laag met 80 neurone en 'n uitvoerlaag met een neuron (wat by verstek deur Matlab geskep word); die parameters wat gebruik word is die volgende:
  - die logaritmiese sigmoïdale oordragfunksie (`logsig`) in beide die versteekte en uitvoerlae;

  - die *Resilient Backpropagation* algoritme (`trainrp`);
  - *gradient descent with momentum* afrigting (`learngdm`);
  - gemiddelde vierkante fout (*mean squared error* (`mse`)) as afrigtingsdoelwit; en
  - die verstekfunksies FIXUNKNOWNS en MAPMINMAX word afgeskakel deur elkeen met {} te vervang. (Sien Bylae A vir die rede hiervoor.)

- ▷ Rig af tot 99% akkuraatheid – dit is tot $mse < 0{,}01$.
- ▷ Stop as 1 000 epogge voltooi is.
- ▷ Die verstekwaarde van die minimum toegelate gradiënt word van 1e-6 na 1e-10 verander. (Verduideliking in Bylae A.)
- ▷ Matlab verdeel die invoerdata by verstek in afrigtings-, validasie- en toetsdata. In 'n vorige studie [Fic02] is bepaal dat oorpassing (memorisering) nie 'n probleem is by die lettergreepverdelingsprobleem nie. Ook in Matlab se gebruikersgids [Mat13] word dit gestel dat die moontlikheid van oorpassing gering is as die aantal parameters in die netwerk (verbindings) baie minder is as die aantal afrigpare wat gebruik word. Ten einde al die data vir afrigting te gebruik, word die verdelingsfunksie afgeskakel (`net.divideFcn='';`).
- ▷ Die netwerk word afgerig (`[net,tr] = train(net,P,T)`) en die gewig- en beladingsmatrikse word as netNEUR.mat gestoor (`save netNEUR net`).

4.3 Ontwikkeling van neurale netwerk

In die afdelings 4.3.1 tot 4.3.4 bespreek ons die proses wat gevolg is om die argitektuur van 'n neurale netwerk vir lettergreepverdeling te bepaal. In afdeling 4.3.5 bespreek ons die bepaling van die venstergrootte wat die beste resultate vir lettergreepverdeling lewer en in afdeling 4.3.6 die bepaling van die optimale neurale netwerk.

Ons toon telkens slegs die akkuraatheid op woord- en verdelingsgeleentheidsvlakke, aangesien die ander maatstawwe tot 'n groot mate daarmee saam fluktueer. Slegs waar die beste drie neurale netwerke vergelyk word, toon ons die volledige resultate.

4.3.1 Aantal versteekte neurone

Ten einde 'n idee te kry van die aantal neurone wat vir optimale resultate in die versteekte laag van 'n tweelaag-neuralenetwerk vir lettergreepverdeling moet wees, het ons neurale netwerke met verskillende getalle versteekte neurone afgerig en getoets.

Ons het 'n willekeurige lys van 50 000 woorde uit die lettergreepverdeelde leksikon onttrek en afrigtingsdata met 4–4-vensterkonfigurasie daaruit gegenereer. Neurale netwerke met 70, 80,

..., 190 versteekte neurone is hiermee afgerig en met die volledige toetsdatastel van ±10 000 woorde getoets. Die saad vir kansgetalgenerering is dieselfde gehou.

Tabel 4.2 toon die akkuraatheid op woordvlak (WV) en verdelingsgeleentheidsvlak (VGV) van hierdie neurale netwerke. Die program D.3.2 is gebruik om neurale netwerke te toets.

Versteekte neurone	**Akkuraatheid**	
	WV	**VGV**
70	0,8650	0,9816
80	0,8665	0,9818
90	0,8562	0,9805
100	0,8558	0,9806
110	0,8837	0,9835
120	0,8629	0,9813
130	0,8765	0,9826
140	0,8610	0,9811
150	0,8718	0,9824
160	0,8819	0,9839
170	**0,9008**	**0,9855**
180	0,8644	0,9815
190	0,8568	0,9806

Tabel 4.2: Akkuraatheid van neurale netwerke afgerig met afrigtingsdata uit 50 000 woorde

Uit hierdie resultate blyk dit dat daar wel 'n toename in akkuraatheid is as meer versteekte neurone gebruik word. Daar is egter geen lineêre verbetering te bespeur nie. Die aantal versteekte neurone om vir die optimale neurale netwerk te gebruik, moes dus empiries bepaal word deur verskeie neurale netwerke met elke konfigurasie af te rig.

Let daarop dat selfs vir relatief min afrigtingsdata (50 000 woorde) goeie verdelingsgeleentheidsvlakakkuraatheid (wat gewoonlik as norm geneem word) verkry is. Dit wissel slegs in die desimale waardes tussen 98,05% en 98,55% oor die hele spektrum van 70 tot 190 versteekte neurone. Op woordvlak wissel die akkuraatheid egter tussen 85,58% en 90,08%.

4.3.2 Hoeveelheid afrigtingsdata

Om te bepaal of meer afrigtingsdata beter resultate lewer, het ons ook neurale netwerke met 4–4-afrigtingsdata uit die volledige afrigtingsdatastel van ±173 000 woorde afgerig. Drie neurale netwerke is vir elke konfigurasie (130, 140, ..., 200 versteekte neurone) afgerig en die akkuraatheid van die beste neurale netwerk vir elkeen word in Tabel 4.3 getoon.

Hieruit blyk dit dat marginaal beter akkuraatheid wel verkry word indien meer afrigtingsdata gebruik word. Die maksimum verdelingsgeleentheidsakkuraatheid vir die neurale netwerk wat met ±173 000 woorde afgerig is, is 98,76% teenoor 98,55% met 50 000 woorde. Ook hier is geen lineariteit tussen die aantal versteekte neurone en akkuraatheid waarneembaar nie.

Versteekte neurone	Akkuraatheid	
	WV	**VGV**
130	0,8874	0,9853
140	0,8987	0,9863
150	0,9005	0,9867
160	0,9033	0,9870
170	0,9068	0,9874
180	0,8947	0,9859
190	**0,9085**	**0,9876**
200	0,8969	0,9862

Tabel 4.3: Akkuraatheid van neurale netwerke afgerig met volledige afrigtingsdatastel

4.3.3 Algoritmes

In 'n vorige studie [Fic02] is bevind dat veerkragtige terugverspreiding (*Resilient Backpropagation* (RP)) die beste algoritme is om vir lettergreepverdeling te gebruik. Volgens Matlab se gebruikersgids is die geskaalde verwante-hellingalgoritme (*Scaled Conjugate Gradient* (SCG)) byna net so vinnig en effektief as RP vir patroonherkenning. Ons het dus die moontlikheid ondersoek om hierdie algoritme vir lettergreepverdeling te gebruik.

In Tabel 4.4 word die prestasie van die RP- en SCG-algoritmes getoon. Beide is met afrigtingsdata uit 50 000 woorde afgerig en die neurale netwerke het beide 'n enkele versteekte laag met 170 neurone.

Algoritme	Akkuraatheid	
	WV	**VGV**
RP	0,9008	0,9855
SCG	0,8508	0,9799

Tabel 4.4: Vergelyking van PR- en SCG-algoritmes

Hierdie resultate toon dat die RP-algoritme beter as die SCG-algoritme presteer en ons het dit vir verdere afrigting gebruik.

4.3.4 Neurale netwerke met meerdere versteekte lae

Ons het ook die moontlikheid oorweeg om neurale netwerke met meer as een versteekte laag vir die lettergreepverdelingsprobleem te gebruik. Die aantal neurone in elk van die lae moes empiries bepaal word. Ons het die datastel met 50 000 woorde gebruik om te bepaal of twee versteekte lae beter resultate lewer. Tabel 4.5 toon die akkuraatheid op woord- en verdelingsgeleentheidsvlak (WV en VGV) van verskillende netwerkkonfigurasies met twee versteekte lae.

Aantal neurone in versteekte laag		**Akkuraatheid**	
1	**2**	**WV**	**VGV**
90	30	0,8510	0,9801
	50	0,8555	0,9801
	70	0,8519	0,9800
100	30	0,8723	0,9825
	50	0,8670	0,9819
	70	0,8801	0,9832
120	30	0,8723	0,9819
	50	0,8643	0,9814
	70	0,8801	0,9813
150	30	0,8677	0,9818
	50	0,8657	0,9818
	70	0,8734	0,9824
170	30	0,8662	0,9822
	50	0,8737	0,9826
	70	**0,8811**	**0,9833**

Tabel 4.5: Akkuraatheid van drielaag-neuralenetwerke (50 000 woorde)

Die akkuraatheid van die tweelaag-neuralenetwerk met 170 versteekte neurone is 0,9008 op woordvlak en 0,9855 op geleentheidsvlak. Nie een van die drielaag-neuralenetwerke wat afgerig is, lewer beter akkuraatheid as dít nie – selfs nie die neurale netwerk met 170–70–1-konfigurasie nie.

Aangesien die afrigtings- en toepassingstyd toeneem soos die kompleksiteit van neurale netwerke toeneem en die prestasie van die veel meer komplekse 170–70–1-neuralenetwerk nie beter is as dié van die 170–1-neuralenetwerk nie, het ons tot die gevolgtrekking gekom dat drielaag-neuralenetwerke nie geskik is vir lettergreepverdeling nie. Ons het dus slegs tweelaag-neuralenetwerke vir die lettergreepverdelingstaak gebruik.

4.3.5 Verskillende venstergroottes

Om die optimale venstergrootte te bepaal is verskillende groottes vergelyk. Die aantal unieke afrigtingsdatapare wat uit die volledige afrigtingsdatastel gegenereer is, word in Tabel 4.6 getoon.

Hieruit sien ons dat die aantal unieke afrigtingsdatapare met 41% toeneem as ons van 'n 4–4-na 'n 5–5-vensterkonfigurasie gaan. Heelwat meer rekenaarkapasiteit is dus vir afrigting met 5–5-vensters nodig is as vir 4–4-vensters. Ons het dan ook vasgestel dat dit 23% langer neem om 'n neurale netwerk wat met 5–5-vensters af te rig as een met 4–4-vensters.

Venstergrootte	**Aantal unieke afrigtingsdatapare**
6 (3–3)	362 438
8 (4–4)	670 967
10 (5–5)	944 639
12 (6–6)	1 164 123

Tabel 4.6: Aantal unieke datapare gegenereer

In Tabel 4.7 word die akkuraatheid op woord- en verdelingsgeleentheidsvlak (WV en VGV) van neurale netwerke met 150 versteekte neurone wat met die volledige afrigtingsdatastel afgerig is, getoon. Ons het lukraak op 150 versteekte neurone besluit, aangesien neurale netwerke met tussen 130 en 190 versteekte neurone die beste prestasie vir afrigting met die volledige afrigtingsdatastel gelewer het.

	Akkuraatheid	
Venstergrootte	**WV**	**VGV**
6 (3–3)	0,8757	0,9841
8 (4–4)	**0,9005**	**0,9867**
10 (5–5)	0,8939	0,9850
12 (6–6)	0,8813	0,9835

Tabel 4.7: Akkuraatheid van neurale netwerke met verskillende venstergroottes

Hieruit blyk dit dus dat simmetriesverdeelde vensters van grootte agt die beste resultate vir lettergreepverdeling lewer.

4.3.6 Finale neurale netwerk

Ten einde 'n finale netwerk- en vensterkonfigurasie te bepaal, het ons tweelaag-neuralenetwerke wat met die volledige afrigtingsdatastel afgerig is, getoets. Ons het neurale netwerke met tussen 150 en 200 versteekte neurone ondersoek en beide simmetriese en skewe vensterkonfigurasies vir venstergroottes agt en tien oorweeg.

Drie verskillende weergawes van elke neurale netwerk is afgerig aangesien die prestasie van neurale netwerke met dieselfde konfigurasie van een stel finale gewigte na 'n ander verskil. Die saad van die kansgetalgenerator is telkens verander sodat die aanvanklike gewigte van poging tot poging verskil. Tabel 4.8 toon die akkuraatheid van die afgerigte neurale netwerke op beide woord- en verdelingsgeleentheidsvlakke (WV en VGV), terwyl die laaste twee kolomme die beste akkuraatheid van elke konfigurasie toon.

Let daarop dat die woordvlakakkuraatheid van al die neurale netwerke tussen 98,47% en 91,33% val, dus 'n variasie van 2,43%, terwyl die verdelingsgeleentheidsvlakakkuraatheid tussen 98,47% en 98,82% val, dus 'n variasie van slegs 0,35%.

Venster-verdeling	Versteekte neurone	Akkuraatheid: Weergawe 1 WV	1 VGV	2 WV	2 VGV	3 WV	3 VGV	Hoogste akkuraatheid WV	Hoogste akkuraatheid VGV
4–4	150	0,8718	0,9853	0,8960	0,9862	0,9005	0,9867	0,9005	0,9867
	160	0,8855	0,9870	0,8955	0,9861	0,9033	0,9870	0,9033	0,9870
	170	0,9068	0,9874	0,8929	0,9860	0,8920	0,9859	0,9068	0,9874
	180	0,8869	0,9852	0,8947	0,9859	0,8932	0,9859	0,8947	0,9859
	190	0,9085	0,9876	0,8931	0,9859	0,8921	0,9854	0,9085	0,9876
	200	0,8969	0,9862	0,8956	0,9862	0,8945	0,9862	0,8969	0,9862
5–3	150	0,8976	0,9864	0,8815	0,9846	0,8996	0,9867	0,8996	0,9867
	160	0,8924	0,9859	0,8953	0,9861	0,9133	0,9882	**0,9133**	**0,9882**
	170	0,8980	0,9864	0,8896	0,9856	0,9119	0,9879	0,9119	0,9879
	180	0,8942	0,9862	0,8922	0,9857	0,8894	0,9853	0,8942	0,9862
	190	0,8931	0,9859	0,9038	0,9870	0,8915	0,9856	0,9038	0,9870
	200	0,9092	0,9877	0,8981	0,9863	0,9043	0,9870	0,9092	0,9877
3–5	150	0,8906	0,9855	0,8895	0,9857	0,8789	0,9844	0,8906	0,9855
	160	0,9054	0,9873	0,8871	0,9851	0,8574	0,9819	0,9054	0,9873
	170	0,9010	0,9868	0,8869	0,9851	0,8863	0,9851	0,9010	0,9868
	180	0,9018	0,9870	0,9004	0,9869	0,8895	0,9857	0,9018	0,9870
	190	0,8895	0,9854	0,8880	0,9855	0,8844	0,9850	0,8895	0,9854
	200	0,8936	0,9860	0,8908	0,9859	0,8802	0,9842	0,8936	0,9860
5–5	150	0,8852	0,9840	0,8939	0,9850	0,8852	0,9840	0,8939	0,9850
	160	0,8965	0,9856	0,9051	0,9867	0,8996	0,9862	0,9051	0,9867
	170	0,8888	0,9864	0,8896	0,9856	0,8945	0,9852	0,8945	0,9852
	180	0,8981	0,9848	0,8912	0,9853	0,8893	0,9848	0,8981	0,9848
	190	0,8779	0,9837	0,8883	0,9846	0,8921	0,9854	0,8921	0,9854
	200	0,9022	0,9863	0,8826	0,9841	0,8935	0,9853	0,9022	0,9863
6–4	150	0,8976	0,9846	0,8827	0,9840	0,8976	0,9846	0,8976	0,9846
	160	0,9021	0,9864	0,8930	0,9853	0,8900	0,9846	0,9021	0,9864
	170	0,8836	0,9843	0,8936	0,9853	0,8872	0,9845	0,8936	0,9853
	180	0,9081	0,9870	0,8919	0,9847	0,9013	0,9860	0,9081	0,9870
	190	0,8858	0,9845	0,8779	0,9835	0,8949	0,9852	0,8949	0,9852
	200	0,8855	0,9842	0,8852	0,9843	0,8954	0,9854	0,8954	0,9854
4–6	150	0,8916	0,9852	0,9015	0,9862	0,8917	0,9853	0,9015	0,9862
	160	0,8944	0,9856	0,9094	0,9876	0,8953	0,9858	0,9094	0,9876
	170	0,8861	0,9845	0,8841	0,9843	0,8954	0,9856	0,8954	0,9856
	180	0,8941	0,9853	0,8940	0,9855	0,8911	0,9852	0,8941	0,9853
	190	0,8956	0,9856	0,9038	0,9868	0,8852	0,9847	0,9038	0,9868
	200	0,9072	0,9869	0,8932	0,9812	0,8724	0,9829	0,9072	0,9869
7–3	150	0,8826	0,9839	0,9045	0,9864	0,8862	0,9842	0,9045	0,9864
	160	0,8807	0,9838	0,8927	0,9851	0,8987	0,9857	0,8987	0,9857
	170	0,8861	0,9843	0,8890	0,9847	0,8827	0,9840	0,8890	0,9847
	180	0,8996	0,9863	0,8904	0,9854	0,8859	0,9845	0,8996	0,9863
	190	0,8860	0,9845	0,8928	0,9854	0,8846	0,9843	0,8928	0,9854
	200	0,8701	0,9824	0,8929	0,9851	0,8801	0,9837	0,8929	0,9851
3–7	150	0,8962	0,9860	0,8829	0,9843	0,8865	0,9848	0,8962	0,9860
	160	0,8865	0,9850	0,9085	0,9873	0,8555	0,9843	0,9085	0,9873
	170	0,8923	0,9855	0,8951	0,9860	0,8743	0,9832	0,8951	0,9860
	180	0,8996	0,9863	0,8904	0,9854	0,8859	0,9845	0,8996	0,9863
	190	0,8802	0,9840	0,8917	0,9854	0,8923	0,9855	0,8923	0,9855
	200	0,9073	0,9872	0,8957	0,9859	0,8823	0,9672	0,9073	0,9872

Tabel 4.8: Akkuraatheid van neurale netwerke afgerig met verskillende vensterkonfigurasies

In Tabel 4.9 word volledige resultate vir die drie beste neurale netwerke vir lettergreepverdeling getoon.

Venster-verdeling	**Versteekte neurone**	**Reg in** C_p	**Reg weg** C_n	**Mis** F_n	**Fout** F_p	**Prestasiemaatstaf** A	P	R	F
		Woordvlak							
5–3	160	9 123	205	241	644	**0,9133**	0,9341	0,9743	0,9537
5–3	200	9 080	206	238	689	0,9092	0,9295	0,9745	0,9514
4–4	190	9 074	204	259	676	0,9085	0,9307	0,9722	0,9510
		Verdelingsgeleentheidsvlak							
5–3	160	25 020	70 939	485	660	**0,9882**	0,9743	0,9810	0,9776
5–3	200	25 021	70 888	484	711	0,9877	0,9724	0,9810	0,9767
4–4	190	24 997	70 902	508	697	0,9876	0,9729	0,9801	0,9765

Tabel 4.9: Uitkomste en prestasie van beste neurale netwerke vir LG-verdeling

Uit hierdie ontleding sien ons dat die tweelaag-neuralenetwerk met 160 versteekte neurone wat met afrigtingsdata met 'n 5–3-konfigurasie afgerig is, die hoogste akkuraatheid op beide woord- en verdelingsgeleentheidsvlakke lewer. Dit verdeel nie alleen die meeste woorde korrek in lettergrepe nie, maar lewer ook die minste woorde met foutiewe verdelings. Ons het dus hierdie neurale netwerk vir verdere ondersoek gebruik.

Die feit dat die 5–3-konfigurasie die beste prestasie lewer, dui moontlik daarop dat meer letters vóór 'n posisie nodig is om te bepaal of 'n woord daar verdeel moet word as ná die posisie.

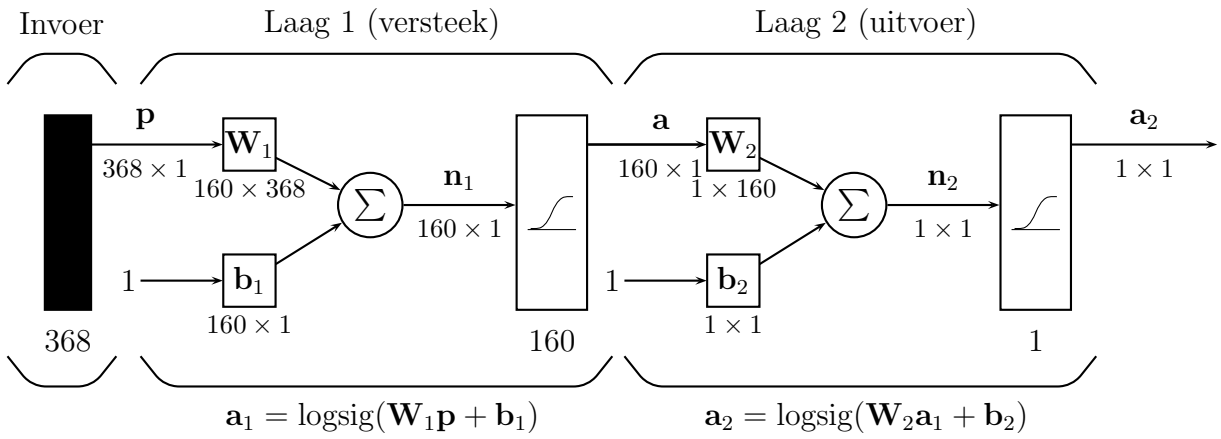
4.3.7 Opsomming

Die neuralenetwerkmodel wat vir lettergreepverdeling gebruik word, is 'n tweelaag vorentoevoer-terugverspreidingsnetwerk (*feedforward backpropagation*) met die volgende eienskappe:

- ▷ Dit bestaan uit 'n versteekte laag met 160 neurone en 'n uitvoerlaag met 'n enkele neuron.
- ▷ Elke invoeritem bestaan uit 368 elemente.
- ▷ 'n Enkele uitvoerneuron is nodig aangesien invoerpatrone bloot as geldige (aangedui deur 'n "1") of ongeldige (aangedui deur 'n "0") verdelingspunte geklassifiseer word.
- ▷ Die logaritmiese sigmoïdale (logsig) oordragfunksie word in die versteekte en uitvoerlae gebruik.

Afrigtingsdata vir die neurale netwerk is met 'n 5–3-vensterkonfigurasie gegenereer en gewigte en beladings is aanvanklik ewekansig gekies.

Figuur 4.7 toon die argitektuur van die neurale netwerk wat vir die lettergreepverdelingsprobleem gebruik word.



Figuur 4.7: Die neurale netwerk vir lettergreepverdeling

4.4 Neurale netwerk vir saamgesteldewoordverdeling

Tydens ontleding van die soorte foute wat voorkom wanneer 'n neurale netwerk aangewend word om lettergreepverdeling te doen, is vasgestel dat foute dikwels by die grense tussen die samestellende dele van saamgestelde woorde voorkom. Die volgende is voorbeelde hiervan (die "!" dui telkens die korrekte koppeltekenposisie aan):

beste-k!opname	liefde-s!teleur-stelling	persoon-s!identiteit	sneeu!g-lans
na-g!ewening	geloof-s!tradisie	handel-s!taal	seru-m!reaksie

Ons sien dus dat foute dikwels voorkom waar die tweede woord in 'n samestelling met 'n klinker begin en waar die verbindings-"s" gebruik word. Ons het dus ook die moontlikheid ondersoek om neurale netwerke te gebruik om saamgestelde woorde eers in hul onderskeie dele te verdeel voordat lettergreepverdeling gedoen word, ten einde hierdie probleem uit te skakel. Die data wat tydens die ontwikkeling van die algoritme wat saamgestelde woorde verdeel gegenereer is (Hoofstuk 3), is vir afrigting en toetsing gebruik.

4.4.1 Ontwikkeling van die neurale netwerk

Soos by lettergreepverdeling is die datastel in afrigtings- en toetsdatastelle verdeel, met ±10 000 woorde wat vir toetsing gereserveer is en die res (±173 000 woorde) wat vir afrigting gebruik is. Ons het ook dieselfde enkodering, venster-teikenpare en prestasiemaatstawwe as voorheen gebruik.

Uit die ontleding by lettergreepverdeling is dit duidelik dat daar geen lineariteit tussen aantal versteekte neurone, hoeveelheid afrigtingsdata en prestasie van neurale netwerke bestaan nie.

Ons het dus neurale netwerke met die volledige afrigtingsdatastel afgerig en slegs een versteekte laag met tussen 150 en 190 versteekte neurone oorweeg. Verder is slegs afrigtingsdata wat met simmetriesverdeelde vensters van groottes 8, 10 en 12 gegenereer is, oorweeg, aangesien daar by lettergreepverdeling so min variasie in die akkuraatheid van die verskillende vensterkonfigurasies was. (Dit was nie moontlik om 'n neurale netwerk vir saamgesteldewoordeverdeling met venstergrootte 6 (3–3) af te rig nie.) Tabel 4.10 toon die vergelyking van die akkuraatheid van hierdie neurale netwerke.

Venster-verdeling	**Versteekte neurone**	**Akkuraatheid: Weergawe**						**Hoogste akkuraatheid**	
		1		**2**		**3**			
		WV	**GV**	**WV**	**GV**	**WV**	**GV**	**WV**	**GV**
4–4	150	0,8547	0,9837	0,8595	0,9842	0,8540	0,9836	**0,8595**	**0,9842**
	160	0,8515	0,9837	0,8554	0,9838	0,8494	0,9830	0,8554	0,9838
	170	0,8571	0,9840	0,8348	0,9809	0,8564	0,9838	0,8571	0,9840
	180	0,8471	0,9827	0,8476	0,9828	0,8437	0,9822	0,8476	0,9828
	190	0,8563	0,9838	0,8434	0,9823	0,8430	0,9822	0,8563	0,9838
5–5	150	0,8477	0,9825	0,8532	0,9833	0,8449	0,9823	0,8532	0,9833
	160	0,8510	0,9831	0,8509	0,9832	0,8474	0,9797	0,8510	0,9831
	170	0,8527	0,9832	0,8547	0,9836	0,8523	0,9833	0,8547	0,9836
	180	0,8575	0,9837	0,8594	0,9842	0,8526	0,9832	**0,8594**	**0,9842**
	190	0,8586	0,9840	0,8532	0,9833	0,8540	0,9834	0,8586	0,9840
6–6	150	0,8489	0,9828	0,8517	0,9832	0,8548	0,9835	0,8548	0,9835
	160	0,8560	0,9837	0,8434	0,9821	0,8515	0,9831	0,8560	0,9837
	170	0,8396	0,9817	0,8449	0,9825	0,8485	0,9828	0,8485	0,9828
	180	0,8465	0,9824	0,8529	0,9831	0,8523	0,9831	0,8529	0,9831
	190	0,8529	0,9831	0,8487	0,9827	0,8508	0,9830	0,8529	0,9831

Tabel 4.10: Akkuraatheid van neurale netwerke vir saamgesteldewoordverdeling

Die volledige resultate vir die twee beste neurale netwerke vir saamgesteldewoordverdeling word in Tabel 4.11 getoon.

Venster-verdeling	**Versteekte neurone**	**Reg in** C_p	**Reg weg** C_n	**Mis** F_n	**Fout** F_p	**Prestasiemaatstaf** A	P	R	F
		Woordvlak							
4–4	150	4 969	3 741	457	967	0,8595	0,8371	0,9158	0,8747
5–5	180	4 895	3 814	592	833	0,8594	0,8546	0,8921	0,8729
		Verdelingsgeleentheidsvlak							
4–4	150	5 885	88 684	516	1 004	0,9842	0,8543	0,9194	0,8856
5–5	180	5 743	88 823	658	865	0,9842	0,8691	0,8972	0,8829

Tabel 4.11: Uitkomste en prestasie van beste neurale netwerke vir saamgesteldewoordverdeling

Alhoewel die NN wat met 4–4-vensters afgerig is hoër akkuraatheid lewer, lewer die NN wat met 5–5-vensters afgerig is 134 minder woorde met foute. Alhoewel dit verdelings in 135 meer woorde mis, wat ons as aanvaarbaar beskou, het ons besluit om die 5–5-NN met 180 neurone as die finale NN vir saamgesteldewoordverdeling te gebruik.

Hoofstuk 5

Beslissingsbome

In hierdie hoofstuk word beslissingsbome as klassifikasietegniek bespreek. Ons begin deur agtergrond oor hierdie tegniek te verskaf waarna die ontwikkeling van 'n beslissingsboommodel vir lettergreepverdeling bespreek word. Die resultate van die optimale boom wat op ons toetsdatastel van $\pm 10\,000$ getoets is, word verskaf en bespreek. 'n Beslissingsboom is ook vir saamgesteldewoordverdeling ontwikkel en getoets. Ook hierdie resultate word bespreek.

5.1 Agtergrond

In beslissingsanalise is 'n beslissingsboom 'n besluitnemingsinstrument wat besluite en hul moontlike gevolge grafiese voorstel. Dit bevat dikwels stogastiese uitkomste en word gebruik om voorwaardelike waarskynlikhede visueel voor te stel en te bereken [Wik13d].

In data-ontginning en masjienleer is 'n beslissingsboom (BB) 'n klassifiseerder in die vorm van 'n boomstruktuur wat data op grond van 'n reeks toetse verdeel. Data wat 'n sekere probleem verteenwoordig, word op grond van toetse op die attribute (veranderlikes of eienskappe) van die probleem verdeel totdat elke voorbeeld in die data aan 'n klas toegeken kan word.

Die volgende twee soorte beslissingsbome word onderskei:

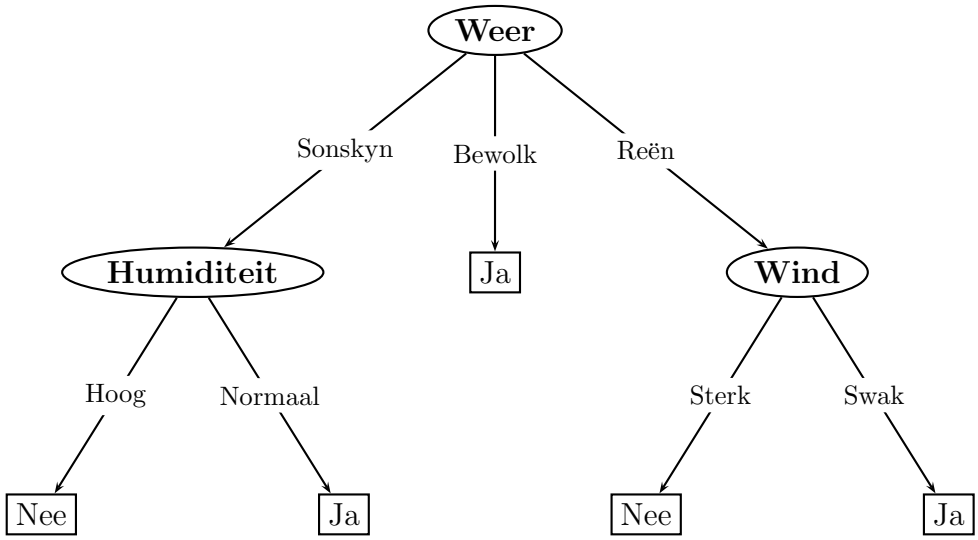
1. *Klassifikasiebome*

   Hier is die klasse waaraan data-items toegeken kan word, diskreet, wat beteken dat 'n data-item óf aan 'n klas behoort, óf nie. 'n Voorbeeld is waar werweldiere op grond van eienskappe soos liggaamstemperatuur, huidbedekking, of dit bene het, geboorte skenk, eiers lê, in water leef, vlieg, hiberneer, ens. as 'n mens, 'n padda, 'n swaan, ens. geklassifiseer word.

2. *Regressiebome*

   Hier is die uitkomste voorspellings wat kontinue waardes aanneem, byvoorbeeld waar eienskappe van 'n huis soos ligging, aantal vertrekke, ens. gebruik word om die prys van 'n huis te voorspel.

In Figuur 5.1 word 'n voorbeeld van 'n klassifikasie-beslissingsboom getoon wat ontwikkel is vir die besluit om te gaan gholf speel, of nie [Qui86]. Dit bestaan uit vertakkingsnodusse waar toetse op attribute (*Weer*, *Humiditeit* en *Wind*) uitgevoer word, vertakkings wat waardes van elke attribuut onder beskouing verteenwoordig (*Sonskyn*, *Reën*, *Sterk*, ens.) en blaar- of eindnodusse waar die klassifikasie gedoen word (*Ja* of *Nee*).



Figuur 5.1: Beslissingsboom vir die besluit om te gaan gholf speel [Qui86]

Deur by die wortel van die boom (*Weer*) te begin en volgens die attribute deur die boom te beweeg, word 'n klas aan elke data-item toegeken.

Een voordeel van beslissingsbome is dat die reëls wat tot 'n sekere uitkoms lei uit die boom afgelees kan word. Dit is dus moontlik om te verduidelik hoe 'n uitkoms bereik is wat soms nodig mag wees, soos om vir iemand wie se aansoek om 'n lening afgekeur is, te kan sê wat die rede is.

5.1.1 Afrigting

Soos voorheen genoem, is dit belangrik om ook vir beslissingsbome voldoende hoeveelhede data in te samel wat die probleem onder beskouing, verteenwoordig. Data wat vir die afrigting van klassifikasiebome geskik is, bestaan uit waarnemings van 'n vaste stel attribute wat die probleem beskryf en die klas (of kategorie) waaraan elke waarneming behoort. Hierdie data verteenwoordig die korrekte gedrag wat van die tegniek verwag word en word gebruik om 'n beslissingsboom af te rig om hierdie gedrag na te boots. Die hoop bestaan dan dat die afgerigte boom sal kan veralgemeen om ook waarnemings wat nie vir afrigting gebruik is nie, korrek te klassifiseer. Dit is belangrik dat die afrigtingsdata alle moontlike klasse sal verteenwoordig en dat genoeg data (gewoonlik honderde of selfs duisende waarnemings) beskikbaar sal wees.

Die afrigtingsdata vir die beslissingsboom in Figuur 5.1 wat die besluit om te gaan gholf speel verteenwoordig, bestaan uit waarnemings oor die weer: hoe die weer lyk, die temperatuur, humiditeit en windsterkte (die attribute) en of die persoon gaan gholf speel, of nie (die uitkoms of klassifikasie). Die data word in Tabel 5.1 gegee. (Let op dat *Temperatuur* geen invloed op die besluit het nie en nie in die beslissingsboom voorkom nie.)

	Attribute				
Item	**Weer**	**Temperatuur**	**Humiditeit**	**Wind**	**Speel**
1	Sonskyn	Warm	Hoog	Lig	Nee
2	Sonskyn	Warm	Hoog	Sterk	Nee
3	Bewolk	Warm	Hoog	Lig	Ja
4	Reën	Matig	Hoog	Lig	Ja
5	Reën	Koel	Normaal	Lig	Ja
6	Reën	Koel	Normaal	Sterk	Nee
7	Bewolk	Koel	Normaal	Sterk	Ja
8	Sonskyn	Matig	Hoog	Lig	Nee
9	Sonskyn	Koel	Normaal	Lig	Ja
10	Reën	Matig	Normaal	Lig	Ja
11	Sonskyn	Matig	Normaal	Sterk	Ja
12	Bewolk	Matig	Hoog	Sterk	Ja
13	Bewolk	Warm	Normaal	Lig	Ja
14	Reën	Matig	Hoog	Sterk	Nee

Tabel 5.1: Data vir die besluit om te gaan gholf te speel of nie

Tydens afrigting moet telkens besluit word op watter attribuut die data verdeel moet word. Verskillende kriteria word gebruik om te bepaal watter attribute vir verdeling gekies moet word. Hierdie kriteria het dit ten doel om die onsekerheid oor die klasse waaraan data-items behoort so gou as moontlik te laat afneem. Verskillende verdelingskriteria word in Afdeling 5.1.1 bespreek.

Die meeste algoritmes wat vir die afrigting van beslissingsbome ontwikkel is, is variasies van die oorspronklike *ID3*-algoritme (*Iterative Dichotomiser 3*) wat deur Quinlan [Qui86] ontwikkel is.

Die *ID3*-algoritme

Hierdie algoritme is ontwerp om beslissingsbome vir probleme met baie attribute en groot datastelle te genereer. Alhoewel dit nie noodwendig die eenvoudigste/kleinste (optimale) boom lewer nie, lewer dit goeie beslissingsbome sonder baie berekenings. Beslissingsbome is dus heuristieke – probleemoplossingsmetodes wat nie noodwendig optimale oplossings lewer nie [Wik13f]. Die algoritme voer 'n gulsige soektog oor die ruimte van moontlike beslissingsbome uit en gebruik die boom wat die gegewe data die beste verdeel [GSM01]. In 'n gulsige soektog word die beste attribuut gekies sonder om vroeëre keuses te heroorweeg. Dit is op "*Occam's razor*" gebaseer, wat in die konteks van beslissingsbome daarop neerkom dat die eenvoudigste

beslissingsboom wat die data onder beskouing klassifiseer waarskynlik die beste sal veralgemeen [HGFO08].

Die *ID3*-algoritme kies telkens die attribuut wat die onsekerheid oor die klas waaraan patrone behoort die meeste sal laat afneem, om op te verdeel. Dit gebruik *Entropie* of *Inligtingswins* (sien Afdeling 5.1.1) om te bepaal watter attribuut by elke vertakking getoets moet word en berus op die volgende twee aannames [Qui86]:

1. 'n Beslissingsboom vir 'n datastel S wat j items van klas J en n items van klas N bevat, sal items in dieselfde verhouding as hul verteenwoordiging in S klassifiseer. Die waarskynlikheid dat 'n arbitrêre item aan klas J behoort, is dus $\frac{j}{j+n}$ en die waarskynlikheid dat dit aan klas N behoort, is $\frac{n}{j+n}$.
2. Aangesien 'n beslissingsboom gebruik word om items te klassifiseer, kan ons dit sien as die bron van 'n boodskap van óf *ja* óf *nee*. Die inligting wat nodig is om die boodskap te genereer, word gegee deur

$$I(j, n) = -\frac{j}{j+n}\log_2\frac{j}{j+n} - \frac{n}{j+n}\log_2\frac{n}{j+n}.$$

As attribuut A met waardes $\{A_1, A_2, \cdots, A_v\}$ as die wortel van die beslissingsboom gebruik word, sal dit S in $\{S_1, S_2, \ldots, S_v\}$ verdeel, sodat S_i die items in S is wat waarde A_i van A bevat. Die verwagte inligting van S_i is $I(j_i, n_i)$ as S_i j_i items van klas J en n_i van klas N bevat. Die verwagte inligting vir die boom met A as wortelnodus is dan die geweegde gemiddelde

$$E(A) = \sum_{i=1}^{v}\frac{j_i + n_i}{j+n}I(j_i, n_i),$$

met die gewig vir die $i$de tak daardie gedeelte van die items in S wat aan S_i behoort. Die inligting wat dus ingewin word as op A verdeel word, is

$$\text{wins}(A) = I(j, n) - E(A).$$

Dit is belangrik om telkens die attribuut te kies wat die meeste inligting sal inwin. Aangesien $I(j, n)$ konstant is vir alle attribute, moet ons dus $E(A)$ minimeer om wins te maksimeer.

Die *ID3*-algoritme ondersoek al die attribute wat vir verdeling beskikbaar is en kies die een wat inligtingswins maksimeer as wortelnodus. Hierdie proses word dan rekursief herhaal om die data te verdeel totdat al die data geklassifiseer word.

Verdelingskriteria

Tydens die afrigting van 'n beslissingsboom moet telkens besluit word watter attribuut getoets moet word en wat die toets moet wees. Bekende verdelingskriteria sluit die volgende in: *Inligtingswins* of *Entropie*; *Twoing*; *Gini-indeks*; die *Chi-kwadraattoets* [HGFO08] en *ReliefF* [KS95]. Breiman [BFOS84] het ook 'n eenvoudige kriterium voorgestel, naamlik om te bepaal hoeveel

gevalle die verskillende toetse verkeerd sal klassifiseer en dan die toets te gebruik wat hierdie waarde sal minimeer.

Ons bespreek vervolgens die verdelingskriteria wat by *Salford Systems* se CART-pakket [Sys13] ingesluit is.

▷ *Inligtingswins*

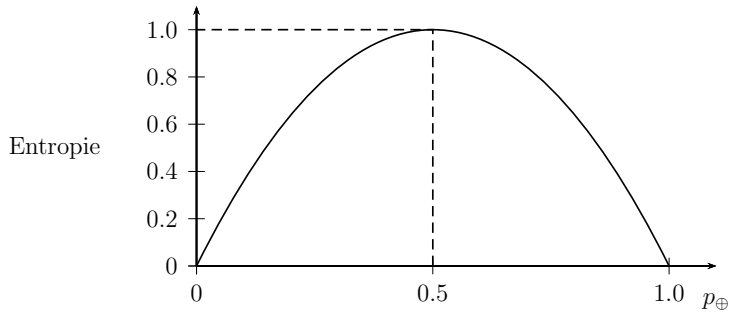
Inligtingswins word aan die hand van *Entropie* gedefinieer wat 'n maatstaf is van die wanorde of onsekerheid in 'n stelsel. In inligtingsteorie is *Entropie* 'n maatstaf van die onsekerheid wat met 'n kansveranderlike geassosieer word. Dit kwantifiseer die inligting wat in 'n boodskap vervat word en word gewoonlik in bisse gemeet.

Die volgorde waarin attribute in 'n beslissingsboom vir verdeling gekies word (en natuurlik ook die toets wat op elke gekose attribuut uitgevoer moet word) moet sodanig wees dat die *afname* in *Entropie* gemaksimeer word.

By 'n binêre klassifikasie waar $p_\oplus$ die gedeelte van positiewe data in datastel S is en $p_\ominus$ die gedeelte negatiewe data, word die *Entropie* van S bereken as

$$Entropie(S) = -p_\oplus \log_2(p_\oplus) - p_\ominus \log_2(p_\ominus).$$

Figuur 5.2 toon die vorm van die *Entropie*-funksie as $p_\oplus$ tussen nul en een varieer.



Figuur 5.2: Die vorm van die *Entropie*funksie

As S suiwer is, met ander woorde al die data is óf positief ($p_\oplus = 1$), óf negatief ($p_\oplus = 0$), is $Entropie(S) = 0$. As S perfek non-homogeen is, met ander woorde daar is net soveel positiewe as negatiewe data-items ($p_\oplus = p_\ominus = 0{,}5$), is $Entropie(S) = 1$ [GSM01]. Entropie moet dus so laag as moontlik wees om onsekerheid te verminder.

In die algemene geval waar k klasse (c_1,c_2,...,c_k) in datastel S voorkom, met gedeelte p_i van S van klas c_i, word die *Entropie* van S bereken as

$$Entropie(S) = -\sum_{i=1}^{k} p_i \log_2(p_i).$$

(Volgens konvensie is $0 * \log_2(0) = 0$.)

Die *Inligtingswins* (IW) van attribuut A is die verwagte afname in *Entropie* as die datastel S op grond van attribuut A verdeel word. *Inligtingswins* word dus gegee deur

$$IW(S,A) = Entropie(S) - \sum_{v \in Waardes(A)} \frac{|S_v|}{|S|} Entropie(S_v),$$

met $|S_v|$ die aantal data-items in S met eienskap v en $|S|$ die totale aantal data-items in S.

▷ *Gini-indeks*

In ekonomiese wetenskappe word die *Gini*-koëffisiënt as 'n maatstaf van die verspreiding van rykdom gebruik. 'n *Gini*-koëffisiënt van nul dui op perfekte verspreiding, met ander woorde almal is ewe ryk (of arm), terwyl 'n *Gini*-koëffisiënt van een daarop dui dat al die rykdom in een persoon se sak is [Dic13].

By beslissingsbome word die *Gini*-indeks as 'n maatstaf van diversiteit in data gebruik. In die algemene geval waar daar k klasse (c_1, c_2, $\ldots$, c_k) in datastel S voorkom, word die *Gini*-ideks bereken as

$$Gini(S) = 1 - \sum_{j=1}^{k} p_j^2,$$

waar p_j die relatiewe frekwensie van klas j in S is.

Vir 'n binêre verdeling wat datastel S in S_1 en S_2 verdeel, word die *Gini*-indeks bereken as

$$Gini(S) = \frac{n_1}{n} Gini(S_1) + \frac{n_2}{n} Gini(S_2),$$

met n_1 en n_2 die aantal data-items in S_1 en S_2 respektiewelik en n die totale aantal data-items in S.

Vir 'n 50:50 verdeling waar eweveel van elke klas in S_1 en S_2 voorkom (dus die slegste senario), is die *Gini*-indeks 0,5. Vir 'n verdeling wat die klasse in die verhouding 70:30 tussen S_1 en S_2 verdeel, is die *Gini*-indeks 0,42. Hieruit is dit duidelik dat wanneer verdeling op die attribuut met die laagste *Gini*-indeks plaasvind, die suiwerste deelversamelings (met die minste variasie) gelewer word.

▷ *Simmetriese Gini-indeks*

Die simmetriese *Gini*-indeks werk met 'n kostematriks. As verskillende kostes nie vir sekere klassifikasiefoute gespesifiseer word nie, is dit identies aan die *Gini*-indeks. Aangesien kostes nie vir spesifieke veranderlikes in die lettergreepverdelingstaak gestel kan word nie, sal hierdie kriterium altyd dieselfde as die *Gini*-indeks wees.

▷ *Klaswaarskynlikheid*

Hierdie is ook 'n variasie van die *Gini*-indeks. *Klaswaarskynlikheid*-beslissingsbome neig om groter as *Gini*-bome te wees en die voorspellings in die eindnodusse is dikwels minder betroubaar. Die detail in die datastruktuur van 'n *Klaswaarskynlikheid*-boom kan dikwels baie waardevol wees.

- *Twoing*

  Die *Twoing*-kriterium streef daarna om die data telkens in twee groepe te verdeel, met 50% verteenwoordiging in elke groep.

- *Geordende Twoing*

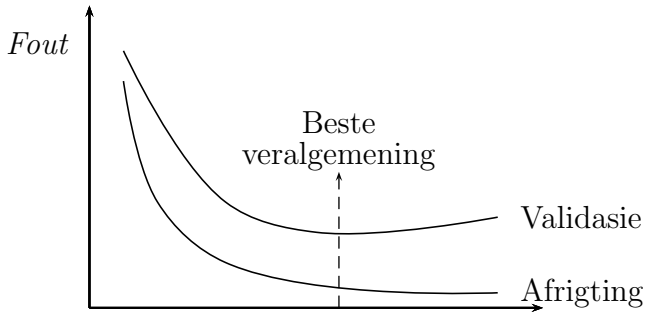
  Hierdie kriterium is bruikbaar wanneer die teikenwaardes geordende klasse is, byvoorbeeld waar data in klasse 1, 2, ..., 5 val. As geordende *Twoing* gebruik word, word klas 4 as meer soortgelyk aan klas 5 en minder soortgelyk aan 1 beskou. Dit sal dus verdelings vermy waar klasse 1 en 5 saamgegroepeer word.

Oorpassing

Oorpassing is 'n algemene probleem by masjienleertegnieke. Wanneer 'n tegniek voluit afgerig word, mag dit die afrigtingsdata memoriseer en sal dit waarskynlik nie in staat wees om te veralgemeen nie.

In 'n boom wat perfek afgerig is, bevat die blaarnodusse dikwels slegs 'n paar (of selfs enkele) data-items. Die toets word dus op 'n baie klein versameling voorbeelde uitgevoer en oorpassing is baie moontlik [Nil96].

In Figuur 5.3 word 'n grafiese voorstelling van oorpassing tydens afrigting gegee. Let op dat die fout op die afrigtingsdata aanhou afneem, maar dat daar 'n punt is waar die fout op validasiedata begin toeneem. By daardie punt sal die model se vermoë om te veralgemeen die beste wees.



Figuur 5.3: Oorpassing

Verskeie benaderings word gebruik om oorpassing in beslissingsbome te oorkom. Hierdie benaderings kan in twee kategorieë opgesom word, naamlik

- om die boom se groei te stop voordat dit perfek is; of
- om die boom toe te laat om voluit te groei en dan party takke terug te snoei.

Laasgenoemde benadering blyk meer suksesvol te wees in die praktyk, aangesien dit moeilik is om te beraam presies wanneer om afrigting te stop. Beide benaderings poog om die optimale boomgrootte te bepaal.

Terugsnoeiing

Tydens die terugsnoeiproses word verdelings en takke wat die minste bruikbaar is, verwyder. Die volgende is voorbeelde van terugsnoeitegnieke wat oor die jare ontwikkel is [Kon98]:

▷ Chi-kwadraatterugsnoeiing (Quinlan, 1982)

Die χ^2-toets word gebruik om te bepaal of dit sinvol is om die attribuut wat as die "beste" gekies is, vir uitbreiding te gebruik, of nie. Dit word egter onbetroubaar wanneer min data-items betrokke is.

▷ Minimum-kostekompleksiteitterugsnoeiing (*Minimum Cost Complexity (MCC)*) (Breiman *et al.*, 1984)

Hierdie metode gebruik 'n *ad hoc* gekose funksie wat die klassifikasiefout en die aantal blaarnodusse in die boom lineêr kombineer. 'n Reeks beslissingsbome word gegenereer en kruisvalidasie word gebruik om die beste boom te kies.

▷ Voorafterugsnoeiing met *ad hoc*-parameters (Kononenko *et al.*, 1984)

Verskeie *ad hoc*-parameters met empiries gekose drempels word gebruik om die uitbreiding van 'n subboom te stop wanneer die aantal data-items of die klassifikasiefout op afrigtingsdata in die huidige nodus die gekose drempel oorskry.

▷ Vorentoeterugsnoeiing met herverdeling (Bratko en Kononenko, 1987)

Data-items in die huidige nodus word telkens in afrigtings- en toetsversamelings herverdeel en 'n enkelvlak subboom word vir elke verdeling gebou. As die gemiddelde klassifikasiefout van die subbome groter as die klassifikasiefout in die huidige nodus is, word die uitbreiding van die subboom gestaak en die huidige nodus word 'n blaarnodus.

▷ Agternaterugsnoeiing met Laplace se wet (Niblett en Bratko, 1986)

Die veralgemeende Lapace-opvolgingswet vir beraming van die waarskynlikheid dat 'n klas C_i die meerderheid in die nodusse sal hê, word gegee deur

$$P(C_i|nodus) = \frac{N(C_i\&nodus) + 1}{N(nodus) + \#klasse},$$

met $N(nodus)$ die totale aantal data-items in die huidige nodus en $N(C_i\&nodus)$ die aantal data-items van klas C_i. Die subboom word teruggesnoei as die beraamde fout van die huidige nodus kleiner as die beraamde fout van die subboom is.

▷ Pessimistiese terugsnoeiing (Quinlan, 1993)

Dit gebruik binomiale vertrouensintervalle om die bogrens van die fout in die huidige nodus te beraam.

▷ Minimum-beskrywingslengte (*Minimum Description Length (MDL)*) terugsnoeiing

Kononenko [Kon98] het 'n eenvoudige MDL-maatstaf ontwikkel om die gehalte van attribute te beraam en 'n deelboom terug te snoei as $\text{MDL}(nodus) \leq 0$.

5.2 Lettergreepverdeling

Ons het CART, wat by Salford Systems se Predictive Mining Suite ingesluit is, gebruik om beslissingsbome vir die lettergreepverdelingstaak af te rig [Sys13]. In die volgende afdelings bespreek ons eers die enkodering wat gebruik is en dan die proses wat gevolg is om 'n beslissingsboom vir lettergreepverdeling te ontwikkel

5.2.1 Enkodering

Soos by neurale netwerke is afrigtingsdata vir lettergreepverdeling gegenereer deur vensters van 'n sekere lengte oor woorde te skuif. Vir elke venster is 'n teikenwaarde verskaf wat aandui of verdeling in 'n bepaalde posisie in die venster geldig is of nie.

Aanvanklik het ons dit oorweeg om dieselfde enkodering as vir neurale netwerke te gebruik, waar elke posisie van 'n venster deur 'n string van 45 nulle en 'n een in die posisie waarin die letter voorkom, voorgestel word. Met die toets van beslissingsbome wat op hierdie manier afgerig is, was die prestasie uiters swak. By nadere ondersoek het ons bevind dat die reëls wat deur die beslissingsboom gegenereer is meestal op die letters wat nié in 'n venster voorkom nie (dié wat deur nulle verteenwoordig word) fokus, eerder as op dié wat wel daar is. Dit was dus nie in staat om te veralgemeen nie.

Ons het dus besluit om die data in terme van die letters en spasies waaruit die vensters bestaan, te enkodeer. Aangesien CART ASCII-enkodering gebruik, terwyl ons in UTF8 werk, het ons die enkodering soos in Tabel 5.2 gebruik. (Program D.4.1.)

Letter	**Kode**	**Letter**	**Kode**	**Letter**	**Kode**	**Letter**	**Kode**
␣	Sp	î	Ik	b	Bb	q	Qq
a	Aa	o	Oo	c	Cc	r	Rr
á	Ak	ó	Oa	d	Dd	s	Ss
ä	Ad	ò	Og	f	Ff	t	Tt
e	Ee	ö	Od	g	Gg	v	Vv
é	Ea	ô	Ok	h	Hh	w	Ww
è	Eg	u	Uu	j	Jj	x	Xx
ë	Ed	ú	Ua	k	Kk	z	Zz
ê	Ek	ü	Ud	l	Ll	=	Kt
i	Ii	û	Uk	m	Mm	’	Af
í	Ia	y	Yy	n	Nn		
ï	Id	ý	Ya	p	Pp		

Tabel 5.2: Letter-enkodering

CART vereis dat onderskeidende name vir veranderlikes gebruik word. Ons het die veranderlikes $L1$, $L2$, $L3$, $L4$, $L5$, $L6$, $L7$ en $L8$ vir 4–4-vensterkonfigurasie toegeken en die teiken deur TT

aangedui. 'n Voorbeeld van 'n afrigtingslêer vir CART word in Tabel 5.3 getoon.

L1	L2	L3	L4	L5	L6	L7	L8	TT
Kk	Hh	Ee	Ii	Dd	Sp	Sp	Sp	0
Aa	Gg	Tt	Ii	Gg	Ee	Sp	Sp	1
Rr	Tt	Jj	Ii	Ee	Sp	Sp	Sp	0
Rr	Hh	Ee	Ii	Dd	Sp	Sp	Sp	0
Ee	Rr	Ii	Nn	Gg	Sp	Sp	Sp	0
Sp	Sp	Sp	Oo	Nn	Vv	Ee	Rr	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Tabel 5.3: CART invoer

Beslissingsbome is met woorde uit die volledige leksikon wat in lettergrepe verdeel is, afgerig. Soos by NNe is 'n toetsdatastel van ±10 000 woorde onttrek (elke 18de woord) en die res (±173 000 woorde) is as afrigtingsdata gebruik.

Vervolgens word verskillende aspekte van afrigting beskou om die afrigtingskriterium en vensterkonfigurasie vir die beste resultate te bepaal.

5.2.2 Ontwikkeling van beslissingsboom

Kriteriumseleksie

Uit die bespreking van die verskillende beskikbare kriteria in Afdeling 5.1.1 is dit duidelik dat die simmetriese *Gini*-indeks vir die lettergreepverdelingsprobleem identies aan die *Gini*-indeks is. Verder is *Geordende Twoing* nie geskik vir die probleem nie, aangesien slegs twee klasse voorkom. Ons het dus die verdelingskriteria *Gini*-indeks, *Twoing*, *Klaswaarskynlikheid* en *Entropie* vir die woordafbrekingsprobleem ondersoek.

Om te bepaal wat die beste verdelingskriterium vir die lettergreepverdelingsprobleem is, is beslissingsbome met afrigtingsdata van 50 000 willekeurig gekose woorde uit die afrigtingsdatastel met elk van hierdie kriteria afgerig en met die toetsdatastel van ±10 000 woorde wat nie deel van die afrigtingsdata is nie getoets.

Tabel 5.4 toon die persentasie posisies waar koppeltekens korrek weggelaat ($TT = 0$) en ingeplaas ($TT = 1$) is, asook die akkuraatheid op verdelingsgeleentheidsvlak as persentasie. Hieruit is dit duidelik dat *Gini*-indeks en *Twoing* identiese resultate lewer. Ons het dus ook *Twoing* vir verdere ondersoek uitgesluit.

Vensterkonfigurasie

Die moontlikheid om skeef-verdeelde vensters van grootte agt te gebruik, is vervolgens ondersoek. Ons het afrigtingsdata uit 50 000 woorde uit die afrigtingsdatastel met 4–4, 3–5 en 5–3 vensterkonfigurasies gegenereer en beslissingsbome met elkeen van die verdelingskriteria afgerig

	Prestasie		
Kriterium	$TT = 0$	$TT = 1$	Akkuraatheid
Gini	97,75	97,41	97,66
Klaswaarskynlikheid	97,28	97,46	97,33
Entropie	97,76	97,39	97,66
Twoing	97,75	97,41	97,66

Tabel 5.4: Kriteriumseleksie

en getoets. In Tabel 5.5 word die akkuraatheid van die verskillende konfigurasies getoon. Let daarop dat ons voortaan akkuraatheid nie as 'n persentasie uitdruk nie, ter wille van eenvormigheid met resultate van neurale netwerke en die TEX-algoritme.

	Gini-indeks			*Klaswaarskynlikheid*			*Entropie*		
Konfigurasie	4–4	3–5	5–3	4–4	3–5	5–3	4–4	3–5	5–3
Aantal nodusse	1 019	1 085	1 008	588	1 827	432	1 080	1 085	1 009
Akkuraatheid	0,9766	0,9760	0,9749	0,9733	0,9762	0,9700	0,9766	0,9760	0,9748

Tabel 5.5: Akkuraatheid van verskillende vensterkonfigurasies

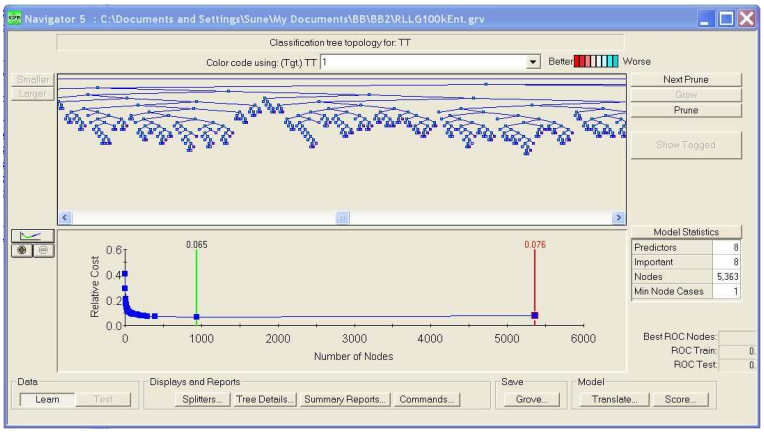
By *Gini*-indeks en *Entropie* lewer die 4–4-vensterkonfigurasie die beste resultate. Ons het dus nie die gebruik van skewe verdelings verder oorweeg nie.

Optimale teenoor groter bome

Om te bepaal of groter bome moontlik beter resultate vir die lettergreepverdelingsproses te gebruik, is die *Entropie*-beslissingsboom wat met afrigtingsdata uit 100 000 willekeurig gekose woorde afgerig is, gebruik. Figuur 5.4 toon die grafiese uitvoer van hierdie afrigting. In die onderste venster, aan die linkerkant van die kostegrafiek word die koste van die optimale boom met 938 nodusse in groen getoon ($koste = 0{,}065$), terwyl die groot boom met 5 363 nodusse (in rooi) aan die regterkant voorkom ($koste = 0{,}076$). In hierdie geval is *koste* die verhouding tussen die klassifikasiesukses en die grootte van die boom.

Beide die optimale en die groter boom is met die toetsdatastel van ±10 000 woorde getoets. 'n Perl-program (Program D.4.3) is hiervoor geskryf wat die beslissingsboomreëls in die afbrekingsproses gebruik. Tabel 5.6 toon die prestasie van die bome volgens die prestasiemaatstawwe Akkuraatheid (A), Presisie (P), Herroeping (R) en die F-telling (F) wat in Afdeling 3.4 bespreek word.

Alhoewel die groot boom beter as die optimale boom presteer, het dit meer as 48 uur geneem om die ±10 000 woorde te verdeel, teenoor die ±8 uur van die optimale boom. Die gemiddeld 17 sekondes wat dit die groot boom neem om 'n woord in lettergrepe te verdeel, is te lank om



Figuur 5.4: Optimale en groter bome

Aantal nodusse	**Reg in** C_p	**Reg weg** C_n	**Mis** F_n	**Fout** F_p	**Prestasie (%)** A	P	R	F
	Woordvlak							
938	8 299	203	208	1 503	0,8325	0,8467	0,9755	0,9100
5 363	8 505	206	522	980	0,8529	0,8967	0,9422	0,9200
	Verdelingsgeleentheidsvlak							
938	24 942	70 010	560	1 592	0,9778	0,9400	0,9780	0,9600
5 363	24 747	70 578	755	1 024	0,9817	0,9603	90,704	0,9700

Tabel 5.6: Vergelyking van optimale boom en groter boom

prakties bruikbaar te wees. Die optimale boom neem slegs 3 sekondes per woord en ons het besluit om slegs optimale bome vir woordafbreking te gebruik.

Veralgemening

Om beslissingsbome se vermoë om te veralgemeen te toets, is beslissingsbome met *Entropie*, *Gini*-indeks en *Klaswaarskynlikheid*-verdelingskriteria afgerig. Die volledige afrigtingsdatalys is vir afrigting gebruik en elke beslissingsboom is met die toetsdata van ±10 000 woorde getoets.

Tabel 5.7 toon die resultate op beide woord- en verdelingsgeleentheidsvlakke. Dit toon die aantal nodusse van elke beslissingsboom en die prestasie van elkeen.

Aangesien *Entropie* die hoogste akkuraatheid en herroeping lewer, het ons besluit dat dit die aangewese kriterium is om vir lettergreepverdeling te gebruik.

Venstergrootte

Ten einde te bepaal of beter resultate moontlik is as kleiner of groter vensters vir afrigting gebruik word, is afrigtingsdata met venstergroottes 6, 8, 10 en 12 gegenereer. *Entropie-*

Verdelings-kriterium	**Aantal nodusse**	**Reg in** C_p	**Reg weg** C_n	**Mis** F_n	**Fout** F_p	**Prestasie (%)** A	P	R	F
		Woordvlak							
Entropie	895	8 375	206	182	1 450	**0,8402**	0,8524	**0,9787**	**0,9100**
Gini	962	8 315	208	260	1 430	0,8345	**0,8533**	0,9697	**0,9100**
Klaswh.	894	7 939	206	251	1 817	0,7975	0,8138	0,9694	0,8800
		Verdelingsgeleentheidsvlak							
Entropie	895	25 005	70 069	499	1 531	**0,9791**	0,9423	**0,9804**	**0,9600**
Gini	962	24 879	70 084	626	1 515	0,9780	**0,9426**	0,9755	**0,9600**
Klaswh.	894	24 873	69 650	631	1 950	0,9734	0,9273	0,9753	0,9500

Tabel 5.7: Resultate vir verskillende verdelingskriteria (volledige afrigtingsdatastel)

beslissingsbome is met hierdie data afgerig en getoets. Die resultate word in Tabel 5.8 getoon.

Venster-konfigurasie	**Reg in** C_p	**Reg weg** C_n	**Mis** F_n	**Fout** F_p	**Prestasie (%)** A	P	R	F
	Woordvlak							
3–3	7 877	207	148	1 981	79,15	79,90	**98,16**	88,09
4–4	8 375	206	182	1 450	**84,02**	**85,24**	97,87	**91,12**
5–5	8 327	205	221	1 460	83,54	85,08	97,41	90,83
6–6	8 228	204	219	1 562	82,56	84,04	97,41	90,23
	Verdelingsgeleentheidsvlak							
3–3	25 040	69 445	465	2 154	97,30	92,08	**98,18**	95,03
4–4	25 005	70 069	499	1 531	**97,91**	**94,23**	98,04	**96,10**
5–5	24 911	70 057	594	1 542	97,80	94,17	97,67	95,89
6–6	24 929	69 965	576	1 634	97,72	93,85	97,74	95,76

Tabel 5.8: Verskillende venstergroottes

Die 4–4-vensterkonfigurasie lewer die beste akkuraatheid, presisie en F-telling. Die *Entropie*-beslissingsboom wat hiermee afgerig is, is dus die optimale boom om vir die lettergreepverdelingstaak te gebruik.

Detail oor die finale beslissingsboom in terme van CART se grafiese platvorm word in Bylae B verskaf.

5.2.3 Bespreking

Dit blyk uit Tabel 5.8 dat die beste beslissingsboom nie baie goed vaar met die lettergreepverdelingstaak nie. Dit maak foute in 14% ($\frac{1\,450}{10\,213}$) van die toetswoorde en mis verdelings in

2%. Dit wil dus voorkom asof die beslissingsboom soveel moontlik koppeltekens inplaas om nie verdelings te mis nie en in die proses baie foute maak.

Die soorte foute wat die beslissingsboom gemaak het, is ontleed. In die uitvoer van die toetsprogram word 'n koppelteken wat verkeerd ingevoeg is deur 'n "∗" aangedui, terwyl 'n koppelteken wat weggelaat is deur 'n "!" aangedui word.

Van die 1 450 woorde waarin foute voorkom, bevat 1 113 woorde (76,8%) koppeltekens weerskante van medeklinkers. Die volgende is onder andere redes hiervoor:

(a) Wanneer die tweede woord in 'n saamgestelde woord met 'n klinker begin, plaas die beslissingsboom die korrekte koppelteken in, maar aangesien die medeklinker gewoonlik saam met 'n klinker na die volgende lettergreep oorgedra word, word 'n koppelteken ook voor die medeklinker ingevoeg, wat natuurlik verkeerd is.

*ve-se*l-in-houd ver-keer*s-on-ge-luk-ke vier-si-lin-de*r-en-jin*

Hierdie probleem kan waarskynlik uitgeskakel word deur saamgestelde woorde eers in die dele waaruit dit saamgestel is, te verdeel voordat lettergreepverdeling gedoen word.

(b) Aangesien onreëlmatigheid by lettergreepverdeling rondom die letter *s* bestaan, blyk dit dat die beslissingsboom nie 'n besluit neem nie, maar eenvoudig koppeltekens weerskante van die *s* invoeg.

*voor-s*ny-mes af-s*kyn-sel ar-beid*s-te-ra-peu-te be-ve*s-ti-ging*

In die afrigtingsdata is die patroon *Oo Oo Rr Ss↑ Nn Yy Mm Ee 0* byvoorbeeld teenwoordig wat die koppelteken ná die *s* in *voorsnymes* behoort te voorkom. (Die ↑ dui op die posisie waarop die teiken berus.) Netso is *Bb Ee Vv Ee ↑ Ss Tt Ii Gg 0* teenwoordig wat die koppelteken vóór die *s* in *bevestiging* behoort te voorkom. Dit is dus nie duidelik hoekom die beslissingsboom hierdie foute maak nie. 'n In-diepte ondersoek na die interaksie tussen die beslissingsboom-reëls sal nodig wees om dit te bepaal, maar dit word nie in hierdie studie gedoen nie.

(c) Ook waar daar as gevolg van taalkonvensies afbrekingsteenstrydighede bestaan, plaas die beslissingsboom bloot koppeltekens weerskante van die teenstrydige letter waar die onsekerheid bestaan.

*boer-boon-t*jie koe-ver*t-jies huur-oo*r-een-kom*s-te in-kom-s*te-be-las-ting*

Die beslissingsboom is duidelik nie in staat om hierdie onderskeid te maak nie. Let op die dubbele fout in *huurooreenkomste*.

Aangesien die verkeerde koppelteken in ±50% van die gevalle vóór en ±50% ná die teenstrydige letter voorkom, is dit nie moontlik om die foute met 'n algemene regstelling te hanteer nie.

Ander lettergreepverdelingsfoute is die volgende:

(a) Wanneer 'n volgende lettergreep met 'n klinker begin, word die medeklinker waarmee die eerste lettergreep eindig bloot na die volgende lettergreep oorgedra.

*kaar*t!ont-le-ding kli*p!uin-tjie noor*d!eu-ro-pe-se an*t!ar*k-ti-ka*

(b) Weereens veroorsaak die letter *s* dikwels foute, soos byvoorbeeld

*ek*s!ten-sief* *duit*s!ta-li-ge* *ho-me-o!s*ta-se* *job*s!ty-ding*

5.3 Beslissingsboom vir saamgesteldewoordverdeling

Ons het die moontlikheid ondersoek om beslissingsbome te gebruik om saamgestelde woorde te verdeel deur die lys van verdeelde saamgestelde woorde vir afrigting te gebruik.

Beslissingsbome is met afrigtingsdata uit die saamgesteldewoordverdeelde leksikon van ±173 000 woorde afgerig en met die toetsdata van ±10 000 woorde getoets om te bepaal hoe goed dit veralgemeen. Tabel 5.9 toon die prestasie (volgens CART) van beslissingsbome wat met verdelingskriteria *Entropie*, *Gini*-indeks en *Klaswaarskynlikheid* afgerig is.

Verdelings-kriterium	**Aantal nodusse**	**Reg in** C_p	**Reg weg** C_n	**Mis** F_n	**Fout** F_p	**Prestasie (%)** A	P	R	F
		Woordvlak							
Entropie	957	2 351	1 777	85	5 923	0,4073	0,2841	**0,9651**	0,4390
Gini	910	2 419	1 932	92	5 793	**0,4194**	**0,2946**	0,9634	**0,4512**
Klaswh.	1 468	2 414	1 827	90	5 805	0,4184	0,2937	0,9641	0,4502
		Verdelingsgeleentheidsvlak							
Entropie	957	6 290	80 471	278	8 920	0,9041	0,4135	0,9577	0,5776
Gini	910	6 277	80 770	291	8 621	**0,9071**	**0,4213**	0,9557	0,5848
Klaswh.	1 468	6 295	80 278	273	8 663	0,9069	0,4208	**0,9584**	**0,5849**

Tabel 5.9: Prestasie van beslissingsbome vir SG-woorde

Gini-indeks lewer die beste prestasie vir saamgestelde woorde. Dit is egter duidelik dat die prestasie swak is, met akkuraatheid van 41,94% en presisie van 29,46% op woordvlak. Op verdelingsgeleentheidsvlak is akkuraatheid 90,71% en presisie slegs 42,13%.

Voorbeelde van foutiewe woordverdeling is die volgende:

*aar*tappel-g*rond* *vreug*d*e-fees* *wet*s-toep*assing* *water-t*afel*
*voet-ganger!r*oe*tes* *vann*ag* *parag*ra*af-vorm* *sne*eu-pop* *gevle*uel*des*

5.4 Gevolgtrekking

Tydens afrigting word 'n toets by elke interne nodus op *een van die veranderlikes (of attribute) uitgevoer*. 'n Beslissingsboom beskou dus een veranderlike op 'n keer en nie 'n string karakters nie. Aangesien lettergreep- en saamgesteldewoordverdeling op patroonherkenning binne karakterstringe berus, is dit verstaanbaar dat beslissingsbome nie goed op hierdie take presteer nie.

Alhoewel die beste lettergreepverdelingsbeslissingsboom akkuraatheid van 97,91% op verdelingsgeleentheidsvlak lewer, is dit slegs 84,02% akkuraat op woordvlak. Aangesien dit lettergreepverdelingsfoute in 14% van die toetswoorde maak, sal ons huiwerig wees om hierdie beslissingsboom in die praktyk te gebruik, aangesien dit potensieel afbrekingsfoute in gedrukte teks sal veroorsaak.

Die resultate vir saamgesteldewoordverdeling dui daarop dat beslissingsbome nie geskik is vir hierdie taak nie. Die beste beslissingsboom is 90,71% akkuraat op verdelingsgeleentheidsvlak, maar slegs 41,94% op woordvlak. Dit maak foute in byna 57% $\left(\frac{5\,793}{10\,213}\right)$ van die toetswoorde. 'n Rede vir hierdie swak prestasie is moontlik dat die afrigtingsdata relatief min koppeltekens bevat en die beslissingsboom hoofsaaklik op veranderlikes wat nie by woordverdelingsposisies betrokke is nie konsentreer.

Hoofstuk 6

Die TEX-algoritme

In hierdie hoofstuk word die woordafbrekingsalgoritme wat TEX gebruik, bespreek. Ons begin deur agtergrond te verskaf oor die TEX-stelsel, waarna ons die algoritme wat vir woordafbreking gebruik word in detail beskryf. Daarna beskryf ons hoe patrone vir lettergreepverdeling gegenereer is en toon die resultate wat daarmee behaal is op ons toetsdatastel van $\pm 10\,000$ woorde. Laastens word patrone ook vir saamgesteldewoordverdeling gegenereer en getoets en die resultate word getoon.

6.1 Agtergrond

TEX is 'n woordverwerkingstelsel wat in die laat 1970s deur Donald Knuth vir Amerikaanse Engels ontwikkel is om veral wiskundige en ander tegnies moeilike dokumente te tik [Knu84]. Daar bestaan dialekte van TEX soos LaTEX, XeTEX, ensovoorts, wat deur middel van makro's aangepas is om dit meer toeganklik en bruikbaar vir gewone gebruikers te maak. Hierdie stelsels lewer dokumente van hoogstaande gehalte en word deur wiskundige en wetenskaplike vaktydskrifte verkies.

TEX is veral gewild omdat die uitleg van dokumente wat dit lewer konsekwent en reg is (as dit natuurlik reg gebruik word). 'n Belangrike aspek daarvan is dat woordspasiëring per paragraaf aangepas word om die uitleg te optimeer. Woordafbreking is een van die tegnieke wat in hierdie proses gebruik word.

In 1983 het Frank Liang [Lia83] 'n woordafbrekingsaloritme vir TEX ontwikkel. In 1991 is die algoritme deur Peter Breitenlohner [LB91] aangepas om 8-bis ASCII-kode in te sluit en in 2001 het David Antoš en Petr Sojka [AS] 'n nuwe weergawe ontwikkel wat ook die gebruik van Unicode (UTF-8) insluit. Ons konsentreer op laasgenoemde weergawe wat die patroongenereringsprogram OPatGen gebruik.

Liang se algoritme, voortaan die TEX-algoritme genoem, is gebaseer op 'n stel patrone wat deur middel van die OPatGen-program uit 'n lys woorde wat in lettergrepe verdeel is, gegenereer word. Indien nodig, word alle moontlike afbreekposisies vir die woorde in 'n paragraaf deur

middel van hierdie patrone bepaal, waaruit die mees geskikte afbreekposisie geselekteer word ten einde die uitleg van die paragraaf te optimeer.

Die doelwit van die TEX-algoritme is om die aantal potensiële afbreekposisies te maksimeer, terwyl foute en stoorspasie geminimeer word. Aangesien dit moeilik, en selfs onmoontlik, is om *alle* potensiële afbreekposisies in 'n afrigtingsdatalys algoritmies te bepaal, is besluit om wel toe te laat dat die algoritme party posisies mis, terwyl foute sover moontlik vermy word [AS].

Die TEX-algoritme hanteer woordafbreking in Engels baie goed, maar vir samestellende tale soos Afrikaans is dit minder suksesvol.

6.2 Woordafbreking in TEX

TEX optimeer die uitleg van elke paragraaf in 'n dokument as 'n eenheid. Besluite oor waar woorde afgebreek moet word, word dus geneem op grond van die beskikbare afbreekposisies in al die woorde in 'n paragraaf [Ant01]. Woordafbreking bestaan uit hoogstens drie fases, naamlik om eers te probeer om die paragraaf sonder enige afbreking in te pas. Indien die oplossing goed genoeg is volgens 'n bepaalde kriterium word die proses gestop, anders word toegelate afbreekpunte vir al die woorde in die paragraaf bepaal, waarmee die beste paragraafuitleg bepaal word. As die oplossing goed is, word die proses gestop, anders word toegelaat dat tussenwoordspasies vergroot word ten einde die beste uitleg te kry.

Die afbreekpunte in 'n woord word bepaal deur die woord eerstens in 'n uitsonderingslys te soek. As dit daar voorkom, is die proses afgehandel, andersins word alle patrone wat met die woord geassosieer word, onttrek en die afbreekpunte word daarvolgens bepaal.

6.3 Patroongenerering

In verskeie tale soos Afrikaans, Duits, Deens, ensovoorts, word letters met diakritiese tekens gebruik om bepaalde uitsprake in teks aan te dui, byvoorbeeld ê, ö, á in Afrikaans. Die Latynse alfabet van 26 letters word met sulke diakritiese letters aangevul, sodat die alfabet vir Afrikaans 43 onderskeibare letters bevat.

Aangesien 'n taal uit duisende woorde bestaan, kom sekere letterpatrone in baie verskillende woorde voor. Die letterpatroon *rte* kom byvoorbeeld in woorde soos *portefeulje*, *huurtermyn*, *soorte*, *vaarteindpunt*, voor. By lettergreepverdeling dui 'n koppelteken die moontlike verdelingsposisies aan, sodat die patroon *r-te* die woorde *soor-te*, *por-tefeulje* en *huur-termyn* hierbo verteenwoordig, terwyl die patroon *rt-e* woorde soos *vaart-eindpunt* en *kwart-eeu* verteenwoordig. Dit is dus nie moontlik om met 'n enkele patroon aan te dui waar die verdeling by *rte* moet plaasvind nie.

In die TEX-algoritme word patrone van verskillende lengtes in verskillende vlakke gegenereer om alle moontlikhede te dek. 'n Geldige verdeling in 'n posisie tussen twee letters word deur

'n ongelyke numeriese waarde (1, 3, ...) aangedui en 'n posisie waar geen verdeling moet plaasvind nie deur 'n gelyke numeriese waarde (2, 4, ...). Die patrone wat gegenereer word, is konteksvry in die sin dat dit op enige plek in 'n woord toegepas kan word.

Tydens patroongenerering word kandidaatpatrone uit die afrigtingsdata gegenereer. Hierdie patrone word geëvalueer en op grond van hul prestasie op die afrigtingsdatastel óf as bruikbare patrone aanvaar, óf verwerp. Die aanvaarde patrone word in 'n datastruktuur (gepakte *trie*-strukture [LB96]) gestoor wat ontwerp is om minimum spasie te gebruik en waaruit patrone vinnig opgeroep kan word. Besonderhede word in Bylae C verskaf.

Vir woordafbreking word woordafbrekings- of lettergreepverdelingspatrone deur die program OPatGen gegenereer. OPatGen is onafhanklik van beide die taal waarvoor dit gebruik word en die toepassingsdomein waarin dit gebruik word [SA].

TEX-patrone bestaan uit deelversamelings van die letterkombinasies uit 'n woord, gekombineer met heelgetalle uit die versameling $\{0, 1, \ldots, 9\}$. Byvoorbeeld, vir die woord *rekenaar* is *1ke*, *ken6aar* en *eke7naar* verteenwoordigende patrone. Om die woord af te breek, word die patrone gekombineer, terwyl die maksimum waarde in elke tussenletterposisie gebruik word. Die patrone lewer dus *re1ke7n6aar* sodat dit as *re-ke-naar* afgebreek word (onewe getalle dui op geldige afbreekposisies). Die verkeerde afbreking *reken-aar* word verhoed deur die patroon *ken6aar*, aangesien ewe getalle op ongeldige afbreking dui.

Om patrone te genereer, word woorde na kleinletters omgeskakel en die begin- en eindpunte word deur punte aangedui. Ons het dus `.a f b r e k i n g.` vir die woord *afbreking*. Aanvanklik het elke tussenletterposisie 'n waarde van nul (`.a0f0b0r0e0k0i0n0g.`).

Kandidaatpatrone wat vir opname in die patroonstruktuur oorweeg word, word in vlakke gegenereer. Dekkingspatrone (*covering patterns*) rondom geldige afbreekpunte word in onewe vlakke gegenereer, terwyl inhiberingspatrone (*inhibiting patterns*) rondom ongeldige afbreekpunte in ewe vlakke gegenereer word.

In Figuur 6.1 toon ons die generering van patrone vir die woord *afbreking*. Kandidaatpatrone word in lopies met patrone van lengtes $2, 3, \ldots$ rondom die afbreekposisies gegenereer. Die *vlaknommer* word telkens aan die posisie onder beskouing toegeken.

Hierdie voorstelling toon uiteraard nie alle moontlike patrone nie. Inhiberende patrone word in élke tussenletterposisie waar daar nie 'n koppelteken voorkom nie, gegenereer. Verder kan lopies vir lengtes van tot nege letters gegenereer word.

Patrone word in 'n orrelpypformasie om elke punt gegenereer. In Figuur 6.2 word Vlak 3-patrone van lengte vyf uit die woord *deur-kosyn* gegenereer en die orrelpypformasie is duidelik waarneembaar.

Uit bogenoemde voorbeelde is dit duidelik dat baie kandidaatpatrone vir elke woord gegenereer word. Nie alle kandidaatpatrone is egter bruikbaar nie. Elke kandidaatpatroon word met die afrigtingsdatalêer vergelyk om te bepaal hoe gereeld dit goed of sleg presteer.

	.	**a**		**f**	**-**	**b**		**r**		**e**	**-**	**k**		**i**		**n**		**g**	**.**	**Patroon**
Vlak 1 (dekking)																				
Lopie 1			0	f	1	b	0													f1b
(lengte 2)	.	a	0	f	1															.af1
					1	b	0	r	0											1br
									0	e	1	k	0							e1k
							0	r	0	e	1									re1
											1	k	0	i	0					1ki
Lopie 2	.	a	0	f	1	b	0													.af1b
(lengte 3)			0	f	1	b	0	r	0											f1br
							0	r	0	e	1	k	0							re1k
									0	e	1	k	0	i	0					e1ki
					1	b	0	r	0	e	1									1bre1
											1	k	0	i	0	n	0			1kin
Vlak 2 (inhibering)																				
	.	**a**		**f**	**-**	**b**		**r**		**e**	**-**	**k**		**i**		**n**		**g**	**.**	**Patroon**
Lopie 1	.	a	2	f	0															.a2f
(lengte 2)			2	f	0	b														2fb
					0	b	2	r	0											b2r
			0	f	0	b	2	0												fb2
							2	r	0	e	0									2re
							0	r	2	e	0									r2e
					0	b	0	r	2											br2
									2	e	0	k	0							2ek
	⋮										⋮									⋮
Lopie 2	.	a	2	f	0	b	0													.a2fb
(lengte 3)			2	f	0	b	0	r	0											2fbr
			0	f	0	b	2	r	0											fb2r
					0	b	2	r	0	e	0									b2re
	.	a	0	f	0	b	2													afb2
							2	r	0	e	0	k	0							2rek
					0	b	0	r	2	e	0									br2e
							0	r	2	e	0	k	0							r2ek
			0	f	0	b	0	r	2											fbr2
									2	e	0	k	0	i	0					2eki
	⋮										⋮									⋮

Figuur 6.1: Patroongenerering vir die woord *afbreking*

Konfigurasie	**.d**	**e**	**u**	**r**	**-**	**k**	**o**	**s**	**y**	**n.**	**Patroon**
3–2		e	u	r	3	k	o				eur3ko
2–3			u	r	3	k	o	s			ur3kos
4–1	.d	e	u	r	3	k					.deur3k
1–4				r	3	k	o	s	y		r3kosy
5–0	.d	e	u	r	3						.deur3
0–5					3	k	o	s	y	n.	3kosyn.

Figuur 6.2: Orrelpypformasie by patroongenerering

Tabel 6.1 toon die kriteria waarvolgens die prestasie van patrone as goed of sleg geklassifiseer word [Ant01].

Vlak	**Prestasie**	
	Goed	**Sleg**
Onewe (dekkend)	Koppelteken stem ooreen met geldige afbreekpunt	Laat koppelteken in ongeldige afbreekpunt toe
Ewe (inhiberend)	Verwyder foutiewe koppelteken uit vorige vlak(ke)	Verwyder korrekte koppelteken uit vorige vlak(ke)
$\sum$	*goeie_telling*	*slegte_telling*

Tabel 6.1: Beoordeling van prestasie

Die prestasietellings *goeie_telling* en *slegte_telling* word saam met die parameterwaardes wat met elke iterasie deur die gebruiker ingevoer word, naamlik *goeie_gewig*, *slegte_gewig* en *drempel* gebruik om te bepaal of 'n patroon as bruikbaar aanvaar of verwerp moet word.

'n Patroon word verwerp indien

$$goeie\_gewig \times goeie\_telling < drempel,$$

terwyl 'n patroon aanvaar word as

$$(goeie\_gewig \times goeie\_telling) - (slegte\_gewig \times slegte\_telling) \geq drempel.$$

Verder word 'n doeltreffendheidsmaatstaf gebruik om op gulsige wyse die mees doeltreffende patrone in elke iterasie te selekteer, met (goeie) doeltreffendheid

$$Dt = \frac{goeie\_telling}{goeie\_telling + \frac{slegte\_telling}{slegte\_dt}}$$

en slegte doeltreffendheid

$$slegte\_dt = \frac{drempel}{slegte\_gewig}.$$

Patrone wat uiteindelik deur die algoritme as geskik vir lettergreepverdeling beskou word, word in 'n effektiewe datastrukture waaruit gegewens maklik onttrek kan word, gestoor. Frank Liang [Lia83] het 'n datastruktuur, naamlik 'n gepakte *trie* vir die woordafbrekingsprobleem ontwikkel. In Bylae C word die ontwikkeling hiervan bespreek.

6.4 Die gebruik van patrone

Om die lettergreepverdelingsposisies in 'n woord te bepaal, word alle patrone wat met die woord ooreenstem uit die datastruktuur onttrek en die verdelingspunte word volgens die grootste waarde in elke tussenletterposisie gedoen.

Om die proses te illustreer, beskou ons die woord *rekenaar*. Die patrone wat daarmee ooreenstem, is die volgende:

.r	2												
		e	3	k									
			3	k	0	e							
		e	0	k	0	e	7	n	0	a	0	a	0	r
						e	4	n	0	a				
													2	r.

Om hieruit die lettergreepverdelingspunte te bepaal, word die maksimum waarde in elke tussenletterposisie bepaal, naamlik

.r	2	e	3	k	0	e	7	n	0	a	0	a	2	r.

Die posisies met onewe waardes dui die lettergreepverdelingsposisies aan, sodat die woord in lettergrepe verdeel word as *re-ke-naar*.

In Figuur 6.3 toon ons die patrone wat vir die woorde *verdeel* en *letterpatrone* onttrek is en die afbreking daarvolgens.

.v		e		r		d		e		e		l.
v	0	e	2	r								
v	0	e	0	r	3							
v	0	e	0	r	0	d	4					
						d	0	e	0	e	4	l
.v	0	e	2	r	3	d	0	e	0	e	4	l.

(a) Ver-deel

.l		e		t		t		e		r		p		a		t		r		o		n		e.
.l	2																							
				t	3	t	0	e																
									2	r	0	p												
											1	p												
															1	t								
																					1	n	0	e
.l	2	e	0	t	3	t	0	e	2	r	1	p	0	a	1	t	0	r	0	o	1	n	0	e.

(b) Let-ter-pa-tro-ne

Figuur 6.3: Patrone en gevolglike afbreking

6.5 Lettergreepverdeling

Om patrone vir Afrikaans te genereer, is dit nodig om 'n vertalingslêer saam te stel wat alle karakters insluit wat in die data waaruit patrone gegenereer word, mag voorkom. Hierdie lêer (`afru.tra`) word in Tabel 6.2 getoon.

1	1			h	H			ö	Ö	\"o	\"O	ý	Ý	\'y	\'Y
a	A			i	I			p	P			z	Z		
á	Á	\'a	\'A	í	Í	\'i		q	Q			'			
ä	Ä	\"a	\"A	ï	Ï	\"i	\'I	r	R			-			
b	B			î	Î	\^i	\"I	s	S			=			
c	C			j	J		\^I	t	T			/			
d	D			k	K			u	U						
e	E			l	L			ú	Ú	\'u	\'U				
é	É	\'e	\'E	m	M			û	Û	\^u	\^U				
è	È	\‘e	\‘E	n	N			ü	Ü	\"u	\"U				
ê	Ê	\^e	\^E	o	O			v	V						
ë	Ë	\"e	\"E	ó	Ó	\'o	\'O	w	W						
f	F			ò	Ò	\‘o	\‘O	x	X						
g	G			ô	Ô	\^o	\^O	y	Y						

Tabel 6.2: Vertalingslêer vir Afrikaans

Die vertalingslêer sluit moontlike skryfmetodes wat in LaTeX gebruik word in. (Sekere variasies, soos `\'{e}`, word hier weggelaat ter wille van spasie.)

Die syfers 1 1 aan die begin dui aan dat eenletter-lettergrepe aan beide die begin en einde van woorde toegelaat word. Letters wat in Afrikaanse woorde mag voorkom, sluit die 26 letters van die alfabet en 17 diakritiese letters in, dus 43 letters. Die volgende karakters is ook ingesluit: (i) die afkappingsteken (') wat in woorde soos *foto's* gebruik word; (ii) die koppelteken (-) wat diskresionêre lettergreepverdelings aandui; (iii) die gelykaanteken (=) wat harde koppeltekens in saamgestelde woorde soos *adjunk-generaal*, *ma-hulle* en *wag-'n-bietjie* aandui en nie weggelaat mag word nie; (iv) die vorentoeskuinsstreep (/) wat in woorde soos *en/of* gebruik word.

Vir lettergreep- en saamgesteldewoordverdeling het ons, soos by neurale netwerke en beslissingsbome, weergawes van ons leksikon van ±183 000 woorde gebruik. In die een is woorde in lettergrepe verdeel en in die ander is saamgestelde woorde in hul samestellende dele verdeel. Dieselfde afrigtings- en toetsdatalyste van ±173 000 en ±10 000 woorde elk is onderskeidelik vir afrigting en toetsing gebruik.

Die OPatgen-program loop in Linux. Ons het 'n ou weergawe van Suse waarop OPatGen suksesvol loop op 'n virtuele masjien geïnstalleer en gebruik. Die volgende opdragreël is gebruik om OPatgen te inisieer:

```
opatgen -u8 LG_Afrig.txt Patrone.0 Patrone.1 afru.tra
```

In hierdie opdragreël roep `opatgen` die program; `-u8` dui aan dat UTF-8 kodering gebruik

word; `LG_Afrig.txt` bevat die afrigtingsdatalys; `Patrone.0` is 'n leë patroonlêer waarmee die algoritme begin; `Patrone.1` is die patroonlêer waarin die gegenereerde patrone geskryf word (die volgende iterasie begin dan met `Patrone.1` en genereer `Patrone.2` sodat telkens op reeds gegenereerde patrone voortgebou word) en `afru.tra` is die vertalingslêer.

Met elke iterasie word die gebruiker gevra om waardes vir sekere parameters in te voer.

- ▷ *hyph_start*, *hyph_finish* (patroonvlakke)
  Ons het een vlak per iterasie gegenereer en het dus vir Vlak 1 die waardes 1 1 ingevoer, 2 2 vir Vlak 2, ensovoorts.
- ▷ *pat_start*, *pat_finish* (patroonlengtes)
  Ons het deurgaans patrone van twee lengtes per iterasie gegenereer. In Vlak 1 het ons patrone van lengtes een en twee gegenereer deur 1 2 in te voer; in Vlak 2 patrone van lengtes twee en drie deur 2 3 in te voer; ensovoorts.
- ▷ *good weight*, *bad weight*, *threshold*
  Ons het by die vroeë iterasies (tot iterasie 4) 1 1 1 vir *goeie_gewig*, *slegte_gewig* en *drempel* ingevoer aangesien die virtuele masjien se beperkte geheue nie aanvanklike hoër waardes kon hanteer nie. Vanaf iterasie vyf af kon ons groter waardes vir *slegte_gewig* gebruik om afbrekingsfoute strenger te beperk.

In Figuur 6.4 word die uitvoer na die eerste iterasie getoon waarin statistiek oor die patroon-genereringsproses verskaf word.

```
396439 good 171600 bad 35499 missed
91.7815 % 39.7279 % 8.21854 %
Count data structure statistics:
  nodes:                          812
  patterns:                       763
  trie_max:                       2714
  current q_max_thresh:           3
Collecting candidates
19 good and 476 bad patterns added (more to come)
finding 397425 good and 171996 bad hyphens
efficiency = 2.31041
Pattern data structure statistics:
  nodes:                          792
  patterns:                       772
  trie_max:                       1808
  current q_max_thresh:           5
  number of different outputs:    21
544 bad patterns deleted
total of 231 patterns at level 1
hyphenate word list <y/n>? y
Writing file pattmp.1

397425 good 171996 bad 34513 missed
92.0097 % 39.8196 % 7.99027 %
```

Figuur 6.4: Uitvoer na eerste iterasie (vlak 8)

Belangrike aspekte van hierdie uitvoer is die statistiek oor die datastruktuur, die aantal patrone wat in hierdie iterasie bygevoeg is, die totale aantal slegte patrone wat in iterasie verwyder is (544) en die aantal Vlak 1-patrone wat behou is (231). Voorbeelde van hierdie patrone is *a1e*, *aë1*, *by1*, ens.

Na elke iterasie het die gebruiker die keuse om te sien hoe die huidige patrone woorde in die

afrigtingsdatastel verdeel. In hierdie geval het die algoritme 397 425 (92,01%) van die woorde reg beoordeel, in 171 996 (39,82%) foute gemaak (koppeltekens op ongeldige posisies) en in 34 513 (7,99%) koppeltekens gemis. Let op dat 'n woord meer as een fout of mis mag bevat sodat die persentasies nie na 100% optel nie.

In Tabel 6.3 word 'n opsomming van die patroongenereringsproses getoon, terwyl Figuur 6.5 die uitvoer na die finale iterasie in die proses toon.

Parameters			**Prestasie**				
Vlak	**Lengtes**	**Goed, Sleg, Drempel**	**Reg**	**Fout**	**Mis**	**Aantal patrone**	**Doel-treffendheid**
1	1–2	1, 1, 1	397 425 92,01%	171 996 39,82%	34 513 8,00%	231	2,31
2	2–3	1, 1, 1	353 654 81,88%	8 053 1,86%	78 284 18,12%	703	41,06
3	3–4	1, 1, 1	427 918 99,07%	24 575 5,69%	4 020 0,93%	2 940	17,33
4	4–5	1, 1, 1	422 525 97,82%	544 0,13%	9 413 2,18%	5 502	551,09
5	5–6	1, 2, 1	431 826 99,97%	975 0,23%	112 0,03%	4 640	434,43
6	6–7	1, 2, 1	431 805 99,97%	20 0,001%	35 0,03%	704	16 608,00
7	7–8	1, 2, 1	431 937 99,9998%	29 0,007%	1 0,0002%	95	14 397,90
8	8–8	1, 3, 1	431 937 99,9998%	3 0,0007%	1 0,0002%	14	33 226,60

Tabel 6.3: Opsomming van patroongenereringsproses

In die finale iterasie is patrone van lengte agt gegenereer waarvan 14 in die datastruktuur opgeneem is. 'n Voorbeeld van so 'n Vlak 8-patroon is `.dor7s8te.` wat die verdeling van die volledige woord *dorste* spesifiseer. Die afrigtingsdatastel is deur die patrone in lettergrepe verdeel met die volgende resultate: 99,9998% van die woorde is reg verdeel, drie woorde bevat koppeltekens op verkeerde plekke en in een woord is 'n koppelteken gemis.

6.5.1 Resultate

Om die prestasie van die gegenereerde stel patrone te toets, is 'n Perl-program (Program D.5.1) gebruik om die ±10 000 woorde in die toetsdatastel in lettergrepe te verdeel. Die program gebruik die Perl-module `TeX::Hyphen` wat deur Jan Pazdziora [Paz13] ontwikkel is om TEX-patrone te gebruik om lettergreepverdeling te doen.

```
Generating a pass with pat_len = 8, pat_dot = 8

431937 good 12 bad 1 missed
99.9998 % 0.00277818 % 0.000231515 %
Count data structure statistics:
  nodes:                          285156
  patterns:                       82183
  trie_max:                       388736
  current q_max_thresh:           3
Collecting candidates
1 good and 82179 bad patterns added (more to come)
finding 431946 good and 12 bad hyphens
efficiency = 33226.6
Pattern data structure statistics:
  nodes:                          1299450
  patterns:                       490718
  trie_max:                       1319603
  current q_max_thresh:           5
  number of different outputs:    537
478398 bad patterns deleted
total of 14 patterns at level 8
hyphenate word list <y/n>? y
Writing file pattmp.8

431937 good 3 bad 1 missed
99.9998 % 0.000694544 % 0.000231515 %
```

Figuur 6.5: Uitvoer na finale iterasie (vlak 8)

Foute wat deur OPatGen se oorspronklike patrone gelewer is, is ontleed en sommige vreemde lettergreepverdelings is geïdentifiseer. Daar was byvoorbeeld gevalle waar 'n koppelteken voor 'n medeklinker aan die einde van 'n woord ingevoeg is, soos *da-ta-ma-trik*s*, *ge-hal-te-wer*k* en *kuns-veld-to*g*, asook gevalle waarin 'n koppelteken na 'n medeklinker aan die begin van die woord ingevoeg is, soos *b*aan-vak*, *m*aand-uit-ga-we* en *r*aar-der*, wat natuurlik nie aanvaarbaar is nie.

In die patroonlêer is patrone soos *2s.*, *2k.*, *.m2*, *.r2* teenwoordig wat sulke lettergreepverdelings behoort te verhoed, maar dit word nie toegepas nie (die rede is onbekend). Deur egter patrone wat die medeklinker en die letter wat telkens daarvóór of daarná voorkom in die patroonlêer in te sluit, byvoorbeeld *k2s.*, *r2k.*, *.m2a*, *.r2a* is hierdie soort foute uitgeskakel.

Dit was ook duidelik dat diakritiese letters nie herken is nie en lettergreepverdeling is dus nie by sulke letters gedoen nie, byvoorbeeld *aäla-wa*, *blin-de-vlieë*, *geïm-pli-seer-de*, ensovoorts. Ons het vasgestel dat dit 'n koderingsprobleem was en het dit reggestel.

Die patroonlêer wat deur OPatGen gegenereer is, bevat 10 190 patrone wat per hand tot 10 238 aangevul is om onaanvaarbare verdelings soos hierbo uit te skakel.

In Tabel 6.4 word die resultate wat met die finale patroonlêer behaal is, getoon. Ons gebruik weereens die prestasiemaatstawwe soos in Afdeling 3.4.

Vir die lettergreepverdelingstaak lewer OPatGen dus uitstekende resultate met akkuraatheid van 97,03% op woordvlak en 99,56% op verdelingsgeleentheidsvlak.

Reg in	**Reg weg**	**Mis**	**Fout**	**Prestasiemaatstaf**			
C_p	C_n	F_n	F_p	A	P	R	F
Woordvlak							
9 703	207	99	204	0,9703	0,9794	0,9899	0,9846
Verdelingsgeleentheidsvlak							
25 282	71 393	223	206	0,9956	0,9919	0,9913	0,9916

Tabel 6.4: Resultate op die toetsdatastel

6.6 Saamgesteldewoordverdeling

Alhoewel lettergreep- en saamgesteldewoordverdelingsposisies saamval en die TEX-algoritme vir lettergreepverdeling ook saamgesteldewoordverdeling behoort te kan doen, maak dit dikwels foute by die grense tussen samestellende dele in woorde wat nie in die afrigtingsdata teenwoordig was nie. Ons het OPATGEN dus ook gebruik om patrone vir saamgesteldewoordverdeling te genereer. Tabel 6.5 toon 'n opsomming van die proses.

Parameters			**Prestasie**				
Vlak	**Lengtes**	**Goed, Sleg, Drempel**	**Reg**	**Fout**	**Mis**	**Aantal patrone**	**Doel-treffendheid**
1	1–2	1, 1, 1	49 282 43,72%	29 004 25,73%	63 435 56,29%	126	1,70
2	2–3	1, 1, 1	37 493 33,26%	1 228 1,09%	75 224 66,74%	550	13,47
3	3–4	1, 1, 1	100 721 89,36%	27 494 24,39%	11 996 10,64%	5 774	3,66
4	4–5	1, 1, 1	94 472 83,81%	243 0,22%	18 245 16,19%	5 743	219,48
5	5–6	1, 1, 1	112 484 99,79%	2 969 2,63%	233 0,21%	8 565	37,68
6	6–7	1, 2, 1	112 430 99,75%	5 0,004%	287 0,25%	1 930	5 622,00
7	7–8	1, 2, 1	112 712 99,996%	6 0,005%	5 0,004%	241	17 271,00
8	8–8	1, 2, 1	112 712 99,996%	1 0,0009%	5 0,0004%	5	–

Tabel 6.5: Opsomming van patroongenereringsproses

Die patrone maak slegs een fout in die saamgesteldewoordverdeelde afrigtingslys, naamlik *legering-staal* wat as *legerings-taal* verdeel word. Woorde soos *programmerings-taal* en *aan-*

biedings-taal het dieselfde letterkombinasie rondom die verdelingspunt, maar met 'n ander verdeling. Die dubbelsinnigheid veroorsaak dus dat die patrone nie die korrekte afbreking kan bepaal nie. Hierdie woord word in 'n uitsonderingslys geplaas.

6.6.1 Resultate

Die prestasie van die gegenereerde patrone op die toetsdatastel is bepaal. Soos voorheen het onaanvaarbare verdelings voorgekom, soos *g-roeselig*, *s-aktyd*, *kroningsee-d* en *self-s* waar enkele medeklinkers aan die begin en einde van woorde afgebreek word.

Hierdie soort foute is soos voorheen reggestel deur patrone per hand in die patroonlêer in te voeg. Die patroonlêer soos deur OpatGen gegenereer bevat 22 010 patrone wat weereens per hand aangevul is tot 22 026.

Tabel 6.6 toon die prestasie van die finale patroonlêer op die toetsdata van ±10 000 woorde.

Reg in	**Reg weg**	**Mis**	**Fout**	**Prestasiemaatstaf**			
C_p	C_n	F_n	F_p	A	P	R	F
Woordvlak							
5 416	3 971	438	309	0,9263	0,9460	0,9252	0,9355
Verdelingsgeleentheidsvlak							
6 170	89 082	524	318	0,9912	0,9510	0,9217	0,9361

Tabel 6.6: Resultate op die toetsdatastel

Die patrone het dus in 309 woorde (3%) woordverdelingsfoute gemaak en verdelings in 438 (4%) woorde gemis. Met akkuraatheid van 99,12% op verdelingsgeleentheidsvlak en 92,63% op woordvlak is dit beslis bruikbaar vir verdere ondersoek.

Hoofstuk 7

Vergelyking van tegnieke

In 'n finale stap van hierdie studie het ons die moontlikheid ondersoek dat 'n algoritme wat uit 'n samestelling van die ondersoekte masjienleertegnieke bestaan, beter resultate as die tegnieke op hul eie sou lewer. Dit was dus nodig om te besluit watter van die tegnieke vir insluiting in so 'n algoritme oorweeg moes word.

In hierdie hoofstuk bespreek ons die werking van elk van die tegnieke met verwysing na die lettergreepverdelingstaak. Daarna word 'n vergelyking van die tegnieke se prestasie op beide lettergreepverdeling en saamgesteldewoordverdeling bespreek en ontleed.

7.1 Werking

7.1.1 Neurale netwerke

Neurale netwerke is deur die werking van die brein geïnspireer en, soos die fisiologie van die brein, is die werking daarvan kompleks en moeilik om te verstaan. Dit word as 'n "*black-box*"-tegniek beskou en mense skram dikwels weg daarvan om dit te gebruik. Die proses waardeur 'n neurale netwerk se uitvoer bepaal word, is nie voor-die-handliggend nie en, net soos dit nie duidelik is hoe die menslike brein by 'n sekere oplossing uitkom nie, is dit nie moontlik om die uitkomste van neurale netwerke op 'n logiese manier te verduidelik nie.

'n Afgerigte neurale netwerk bestaan uit 'n stel gewigte waarmee 'n uitvoer deur eenvoudige berekenings bepaal word. Vir die lettergreepverdelingstaak bestaan die optimale neurale netwerk uit 368 invoergewigte, 'n 160×368 matriks van versteektelaaggewigte, 160 versteektelaagbeladings, 160 uitvoerlaaggewigte en een uitvoerlaagbelading. So 'n stel gewigte benodig ± 30 kilogrepe stoorspasie.

Tydens die lettergreepverdelingstaak bereken die neurale netwerk met behulp van die verbindingsgewigte 'n waarde vir elke verdelingsgeleentheid in 'n woord. Die verstekwaarde waarvolgens besluit word of 'n koppelteken op 'n bepaalde posisie ingeplaas moet word, is 0,5. Vir

waardes groter of gelyk aan as 0,5 word 'n koppelteken in die posisie geplaas, en vir waardes kleiner as 0,5 nie.

Hier volg voorbeelde van waardes wat die neurale netwerk by geldige, foutiewe (∗) en gemiste (!) verdelingsposisies gelewer het:

- aan-(1,00)de-(1,00)le-(0,92)por-(1,00)te-(0,99)feul-(1,00)je → *aan-de-le-por-te-feul-je*

  Aangesien al die waardes bo 0,9 is, bestaan daar byna geen twyfel of die verdelings geldig is nie.

- bek-(0,99)ken-(0,95)g*(0,63)or-(0,99)del → *bek-ken-g-ordel*

  In hierdie geval word 'n fout in die posisie tussen *g* en *o* gemaak. Die neuralenetwerkwaarde in hierdie posisie is 0,63 wat hoër as 0,5 is, maar baie laer as dié van die korrekte verdelingsposisie (0,95). In sulke gevalle waar koppeltekens weerskante van 'n medeklinker (anders as 'n *s*) voorkom, mag dit moontlik wees om foute te vermy deur die hoogste waarde as verdelingspunt te kies. 'n Ontleding van sulke situasies sal nodig wees voor so 'n reël geïmplementeer word.

- duw-(0,99)wel-(0,94)tjie!(0,38)s*(0,56)tin-(1,00)gels → *duw-wel-tjies-tin-gels*

  In die hierdie geval is die neuralenetwerkwaarde in die gemiste posisie laag (0,38) terwyl die waarde in die foutiewe posisie (0,56) nie beduidend meer as 0,5 is nie.

- ge-(1,00)voel*(0,89)s!(0,47)kri-(1,00)tiek → *ge-voel-skri-tiek*

  In hierdie geval is die waarde in die gemiste posisie naby 0,5, terwyl die waarde in die foutiewe posisie net onder 0,9 is.

In die laaste twee gevalle is foute rondom die letter *s* gemaak wat die onsekerheid oor sulke verdelings illustreer. Die waardes by die geldige verdelingspunte is telkens onder 0,5 en die waardes in die foutiewe verdelingspunte is nie bo 0,9 nie. Dit mag moontlik wees om sulke foute uit te skakel as verdelings by waardes kleiner as 0,9 by die letter *s* vermy word. Diepgaande ontleding van sulke foute sal egter gedoen moet word voordat so 'n besluit geneem kan word.

7.1.2 Beslissingsbome

Beslissingsbome verdeel data op grond van verdelingskriteria sodat die onsekerheid oor die klas waaraan 'n element behoort, afneem soos in die boom afbeweeg word. By elke vertakking kan die reël wat daar gebruik word van die boom afgelees word. Vir groot beslissingsbome soos die een vir lettergreepverdeling is dit egter nie moontlik om die volledige boom met een opslag op die rekenaar te sien nie, wat dit nie moontlik maak om dit fisies te doen nie.

Die optimale beslissingsboom vir lettergreepverdeling bestaan uit 895 nodusse, wat beteken dat daar 895 verdelingsreëls is wat ±1,5 megagrepe stoorspasie benodig. Die volgende is byvoorbeeld die reëls wat by die tweede en agste nodusse van hierdie BB geld.

```
#/*Terminal Node 2*/
elsif (($L4 eq "Ad" | $L4 eq "Ak" | $L4 eq "Bb" | $L4 eq "Cc" | $L4 eq "Dd" | $L4 eq "Ed" | $L4 eq "Eg" | $L4 eq "Gg" | $L4 eq "Hh" | $L4 eq "Ia" |
        $L4 eq "Id" | $L4 eq "Jj" | $L4 eq "Kk" | $L4 eq "Ll" | $L4 eq "Mm" | $L4 eq "Nn" | $L4 eq "Oa" | $L4 eq "Og" | $L4 eq "Pp" | $L4 eq "Qq" |
        $L4 eq "Rr" | $L4 eq "Tt" | $L4 eq "Ud" | $L4 eq "Vv" | $L4 eq "Ww" | $L4 eq "Ya" | $L4 eq "Zz" ) &
       ($L3 eq "Ad" | $L3 eq "Af" | $L3 eq "Ak" | $L3 eq "Bb" | $L3 eq "Cc" | $L3 eq "Dd" | $L3 eq "Ea" | $L3 eq "Ek" | $L3 eq "Ff" | $L3 eq "Gg" |
        $L3 eq "Hh" | $L3 eq "Ia" | $L3 eq "Kk" | $L3 eq "Kt" | $L3 eq "Oa" | $L3 eq "Od" | $L3 eq "Ok" | $L3 eq "Pp" | $L3 eq "Qq" | $L3 eq "Sp" |
        $L3 eq "Ss" | $L3 eq "Tt" | $L3 eq "Ua" | $L3 eq "Ud" | $L3 eq "Uk" | $L3 eq "Vv" | $L3 eq "Xx" ) &
       ($L5 eq "Aa" | $L5 eq "Ea" | $L5 eq "Ee" | $L5 eq "Eg" | $L5 eq "Ek" | $L5 eq "Ff" | $L5 eq "Gg" | $L5 eq "Ii" | $L5 eq "Jj" | $L5 eq "Oo" |
        $L5 eq "Tt" | $L5 eq "Uu" | $L5 eq "Yy" ) &
       ($L6 eq "Ad" | $L6 eq "Af" | $L6 eq "Ak" | $L6 eq "Bb" | $L6 eq "Cc" | $L6 eq "Dd" | $L6 eq "Ed" | $L6 eq "Ff" | $L6 eq "Gg" | $L6 eq "Hh" |
        $L6 eq "Id" | $L6 eq "Kk" | $L6 eq "Kt" | $L6 eq "Ll" | $L6 eq "Mm" | $L6 eq "Nn" | $L6 eq "Oa" | $L6 eq "Od" | $L6 eq "Pp" | $L6 eq "Qq" |
        $L6 eq "Rr" | $L6 eq "Sp" | $L6 eq "Ss" | $L6 eq "Tt" | $L6 eq "Ua" | $L6 eq "Ud" | $L6 eq "Uk" | $L6 eq "Vv" | $L6 eq "Ww" | $L6 eq "Xx" |
        $L6 eq "Zz"))
{
    $class = 0;
}
#/*Terminal Node 8*/
elsif (($L4 eq "Aa" ) &
       ($L5 eq "Ee" ) &
       ($L3 eq "Ff" | $L3 eq "Gg" | $L3 eq "Hh" ) &
       ($L6 eq "Ll" | $L6 eq "Vv" | $L6 eq "Ww" ) &
       ($L2 eq "Aa" | $L2 eq "Cc" | $L2 eq "Ee" | $L2 eq "Ff" | $L2 eq "Ii" | $L2 eq "Kt" | $L2 eq "Oo" | $L2 eq "Sp" ) &
       ($L7 eq "Ad" | $L7 eq "Af" | $L7 eq "Ak" | $L7 eq "Bb" | $L7 eq "Cc" | $L7 eq "Dd" | $L7 eq "Ea" | $L7 eq "Ed" | $L7 eq "Ee" | $L7 eq "Eg" |
        $L7 eq "Ek" | $L7 eq "Ff" | $L7 eq "Gg" | $L7 eq "Hh" | $L7 eq "Ia" | $L7 eq "Id" | $L7 eq "Jj" | $L7 eq "Kk" | $L7 eq "Kt" | $L7 eq "Ll" |
        $L7 eq "Mm" | $L7 eq "Nn" | $L7 eq "Oa" | $L7 eq "Od" | $L7 eq "Ok" | $L7 eq "Pp" | $L7 eq "Qq" | $L7 eq "Rr" | $L7 eq "Sp" | $L7 eq "Ss" |
        $L7 eq "Tt" | $L7 eq "Ua" | $L7 eq "Ud" | $L7 eq "Uk" | $L7 eq "Uu" | $L7 eq "Vv" | $L7 eq "Ww" | $L7 eq "Xx" | $L7 eq "Ya" | $L7 eq "Zz"))
{
    $class = 1;
}
```

Reël 2 sê dat indien die vierde letter van 'n woord een van 27 letters is, die derde letter een van 27 ander letters, die vyfde letter een van 13 letters en die sesde letter een van 31 letters is, moet daar nie 'n koppelteken ingeplaas word nie. Alhoewel dit duidelik 'n tydrowende taak sal wees, sou dit wel moontlik wees om die stel reëls wat 'n sekere uitkoms lewer, te verskaf.

7.1.3 Die TEX-algoritme

Die TEX-algoritme gebruik 'n stel patrone wat deur die OPatGen-program gegenereer is om afbreekposisies te bepaal. Die patrone word in vlakke gegenereer terwyl die lengte van patrone toeneem soos die vlakke toeneem. Die syfer wat aan tussenletterposisies toegeken word – die vlaknommer – kan dus as 'n annotasie van so 'n posisie beskou word.

Vir 'n bepaalde woord word alle patrone wat met die woord ooreenstem uit die databasis onttrek en die afbreking word dan volgens die grootste waarde in elke tussenletterposisie gedoen. Die patrone word in 'n baie effektiewe datastruktuur gestoor waarin dit maklik opgespoor kan word. Die proses om verdelingspunte te bepaal, behels eenvoudig om die maksimum waarde in elke tussenletterposisie te bepaal en dan op grond van ewe of onewe waardes diskresionêre verdelingspunte toe te ken.

Langer patrone wat in hoër vlakke gegenereer word, het meer invloed op die posisie waar 'n verdeling moet plaasvind. Vir die woord *rekenaar*, byvoorbeeld, word die patrone `e4na` en `eke7naar` uit die databasis onttrek. Aangesien die hoogste waarde 'n onewe getal is, word die woord dus korrek in die posisie voor die n verdeel, naamlik *reke-naar*.

7.2 Prestasie

7.2.1 Woordafbreking

In Tabel 7.1 word die prestasie van die optimale neurale netwerk, beslissingsboom en die TEX-algoritme vir lettergreepverdeling in dalende volgorde getoon.

Tegniek	**Reg in** C_p	**Reg weg** C_n	**Mis** F_n	**Fout** F_p	**Prestasie (%)** A	P	R	F
	Woordvlak							
TEX-algoritme	9 703	207	99	204	0,9703	0,9794	0,9899	0,9846
Neurale Netwerke	9 123	205	241	644	0,9133	0,9341	0,9743	0,9537
Beslissingsbome	8 375	206	182	1 450	0,8402	0,8524	0,9787	0,9112
	Verdelingsgeleentheidsvlak							
TEX-algoritme	25 282	71 393	223	206	0,9956	0,9919	0,9913	0,9916
Neurale Netwerke	25 020	70 939	485	660	0,9882	0,9743	0,9810	0,9776
Beslissingsbome	25 005	70 069	499	1 531	0,9791	0,9423	0,9804	0,9610

Tabel 7.1: Vergelyking van tegnieke vir LG-verdeling

Die TEX-algoritme lewer die beste prestasie by lettergreepverdeling, met 99,56% akkuraatheid op verdelingsgeleentheidsvlak en 97,03% op woordvlak. Dit maak foute in 204 woorde deur koppeltekens op ongeldige posisies in te voeg en dit mis koppeltekens in slegs 99 woorde. Dit behoort dus goeie afbrekingsresultate te lewer sonder om ander tegnieke by te bring.

Die neurale netwerk presteer ook goed met akkuraatheid van 91,33% en 98,82% op woord- en verdelingsgeleentheidsvlak, respektiewelik. Die neurale netwerk alleen maak veel meer foute – 644 woorde (6,3%) woorde met foute en 241 woorde met verdelings gemis. Ons sal dus huiwerig wees om dit op sy eie vir woordafbreking te gebruik en sal die moontlikheid ondersoek om dit met ander tegnieke te kombineer.

Die beslissingsboom presteer die swakste met slegs 84,02% en 97,91% akkuraatheid op woord- en verdelingsgeleentheidsvlak, respektiewelik. Dit maak foute in 1 450 woorde en mis verdelings in 182 woorde. Die BB behoort dus met omsigtigheid vir lettergreepverdeling gebruik te word, of moet selfs uitgeskakel word as moontlike verdelingstegniek.

7.2.2 Saamgesteldewoordverdeling

Tabel 7.2 word die prestasie van die drie tegnieke vir saamgesteldewoordverdeling getoon.

Tegniek	Reg in C_p	Reg weg C_n	Mis F_n	Fout F_p	Prestasie (%) A	P	R	F
	Woordvlak							
TEX-algoritme	5 416	3 971	438	309	0,9263	0,9460	0,9252	0,9355
Neurale Netwerke	4 969	3 741	457	967	0,8595	0,8371	0,9158	0,8747
Beslissingsbome	2 419	1 932	92	5 793	0,4194	0,2946	0,9634	0,4512
	Verdelingsgeleentheidsvlak							
TEX-algoritme	6 170	89 082	524	318	0,9912	0,9510	0,9217	0,9361
Neurale Netwerke	5 885	88 684	516	1 004	0,9842	0,8543	0,9194	0,8856
Beslissingsbome	6 277	80 770	291	8 621	0,9071	0,4213	0,9557	0,5848

Tabel 7.2: Vergelyking van tegnieke vir saamgesteldewoordverdeling

Ook hier lewer die TEX-algoritme uitstekende resultate, met akkuraatheid van 92,63% op woordvlak en 99,12% op verdelingsgeleentheidsvlak. Weereens is dit dus bruikbaar vir saamgesteldewoordverdeling.

Die resultaat van 85,95% akkuraatheid op woordvlak vir die neurale netwerk is relatief laag in vergelyking met die TEX-algoritme, wat daarop dui dat mens dit met omsigtigheid vir saamgesteldewoordverdeling sal gebruik.

Die beslissingsboom se akkuraatheid van 41,94% op woordvlak is onaanvaarbaar laag en ons sal dit nie vir insluiting by 'n gekombineerde algoritme oorweeg nie.

Hoofstuk 8

Gekombineerde algoritme

In die vorige hoofstukke het ons die masjienleertegnieke neurale netwerke, beslissingsbome en die TEX-algoritme in diepte bestudeer. Die tegnieke is vir beide lettergreep- en saamgesteldewoordverdeling aangewend en die resultate is ontleed.

In hierdie hoofstuk ondersoek ons die moontlikheid dat 'n kombinasie van hierdie tegnieke beter lettergreepverdeling sal lewer. Vir lettergreepverdeling gebruik ons die beste neurale netwerk, beslissingsboom en die TEX-patroonlêer (voorgestel deur NNLG, BBLG en OPLG respektiewelik) en vir saamgesteldewoordverdeling die beste neurale netwerk en die TEX-patroonlêer (NNSG en OPSG).

8.1 Ontwikkeling

Die samestellende aard van Afrikaans het tot gevolg dat afbrekingsfoute dikwels by die grense tussen die samestellende dele van saamgestelde woorde voorkom. Ten einde sulke afbrekingsfoute te beperk, ondersoek ons die moontlikheid om lettergreepverdelingsresultate te verbeter deur woorde eers in hul samestellende dele te verdeel voordat dit in lettergrepe verdeel word. Aangesien die neurale netwerk en die beslissingsboom se resultate heelwat swakker is as dié van die TEX-algoritme, gebruik ons NNSG slegs in kombinasie met OPSG vir saamgesteldewoordverdeling, en NNLG en BBLG slegs in kombinasies wat OPLG insluit vir lettergreepverdeling.

Vir die gekombineerde algoritme (K-algoritme) oorweeg ons verskillende kombinasies van hierdie modelle. Ons dui byvoorbeeld die geval waar OPSG en NNSG se uitvoere gekombineer word (+), die resultaat aangestuur (→) word om deur OPLG, NNLG en BBLG in lettergrepe verdeel en die resultate gekombineer word (+) aan as

$$\text{OPSG} + \text{NNSG} \rightarrow \text{OPLG} + \text{NNLG} + \text{BBLG}.$$

Wanneer twee of drie tegnieke se uitvoere gekombineer word, word elkeen se uitvoer bepaal en die uitvoere word met mekaar vergelyk. Slegs wanneer 'n koppelteken in 'n bepaalde posisie in al die uitvoere ooreenstem, word dit in die gekombineerde uitvoer opgeneem. Veronderstel

byvoorbeeld die OPSG-uitvoer vir die woord *stiefoupa* is *stief-oupa* en die NNSG-uitvoer is *stief-ou-pa*. Wanneer dit gekombineer word, is die uitvoer *stief-oupa*. By lettergreepverdeling van die woord *rondomtalie*, byvoorbeeld, is die OPLG-uitvoer *ron-d-om-ta-lie*, die NNLG-uitvoer is *rond-om-ta-lie* en die BBLG-uitvoer is *ron-d-om-t-a-lie*. Wanneer die drie uitvoere gekombineer word, is die resultaat *rond-om-ta-lie*, aangesien die koppelteken voor die *d* en na die *t* nie by al die uitvoere voorkom nie.

Die volgende kombinasies word ondersoek:

- ▷ OPSG + NNSG → OPLG + NNLG + BBLG;
- ▷ OPSG + NNSG → OPLG + NNLG;
- ▷ OPSG + NNSG → OPLG;
- ▷ OPSG → OPLG + NNLG + BBLG;
- ▷ OPSG → OPLG + NNLG;
- ▷ OPSG → OPLG;
- ▷ OPLG + NNLG.

8.1.1 Resultate

Tabel 8.1 toon die resultate van die gekombineerde algoritmes wat met die toetsdatastel van ±10 000 woorde (met ±97 000 verdelingsgeleenthede) getoets is. Die akkuraatheid van die verskillende algoritmes (ALG) op woordvlak (WV) en op verdelingsgeleentheidsvlak (VGV) word getoon, terwyl die akkuraatheid van OPLG, NNLG en BBLG op hul eie ingesluit word vir volledigheid.

	Kombinasie	**WV**	**VGV**
	Slegs OPLG	**0,9703**	**0,9956**
	Slegs NNLG	0,9133	0,9882
	Slegs BBLG	0,8402	0,9791
ALG1	OPSG + NNSG → OPLG + NNLG + BBLG	0,7832	0,9759
ALG2	OPSG + NNSG → OPLG + NNLG	**0,9524**	**0,9942**
ALG3	OPSG + NNSG → OPLG	**0,9544**	**0,9943**
ALG4	OPSG → OPLG + NNLG + BBLG	0,7814	0,9756
ALG5	OPSG → OPLG + NNLG	**0,9628**	**0,9954**
ALG6	OPSG → OPLG	0,9466	0,9934
ALG7	OPLG + NNLG	0,9310	0,9917

Tabel 8.1: Akkuraatheid van versillende kombinasies

Uit die tabel blyk dit dat OPLG op sy eie steeds die beste resultate lewer. Verder is dit duidelik dat die akkuraatheid op woordvlak telkens tot onder 80% afneem as BBLG by die kombinasie

ingesluit word. Dit bevestig ons vorige waarneming dat BBe nie vir woordafbreking gebruik behoort te word nie.

Om 'n geheelbeeld te kry, beskou ons die volledige resultate van die beste presteerders, naamlik OPLG op sy eie, ALG5, ALG3 en ALG2 met prestasie in dalende volgorde, soos in Tabel 8.2 getoon.

	Kombinasie	**Reg in** C_p	**Reg weg** C_n	**Mis** F_n	**Fout** F_p	**Prestasie (%)** A	P	R	F
		Woordvlak							
	Slegs OPLG	9 703	207	99	204	97,03	97,94	98,99	98,46
ALG5	OPSG → OPLG + NNLG	9 625	208	307	73	0,9628	0,9925	0,9691	0,9807
ALG3	OPSG + NNSG → OPLG	9 539	208	78	388	0,9544	0,9609	0,9919	0,9762
ALG2	OPSG + NNSG → OPLG + NNLG	9 519	208	415	71	0,9524	0,9926	0,9582	0,9751
		Verdelingsgeleentheidsvlak							
	Slegs OPLG	25 282	71 393	223	206	0,9956	0,9919	0,9913	0,9916
ALG5	OPSG → OPLG + NNLG	25 136	71 526	369	73	0,9954	0,9971	0,9855	0,9913
ALG3	OPSG + NNSG → OPLG	25 336	71 214	166	388	0,9943	0,9849	0,9935	0,9892
ALG2	OPSG + NNSG→ OPLG + NNLG	25 013	71 531	489	71	0,9942	0,9972	0,9808	0,9889

Tabel 8.2: Vergelyking van algoritmes vir LG-verdeling

Alhoewel OPLG op sy eie die hoogste akkuraatheid op woordvlak lewer, maak dit foute in 204 woorde teenoor die 73 en 71 foute van ALG5 en ALG2, respektiewelik. Aan die ander kant mis OPLG op sy eie lettergreepverdelings in slegs 99 (1%) woorde, terwyl ALG5 verdelings in 307 (3%) woorde mis en ALG2 415 (4%).

Die doelwit is om afbrekingsfoute in gedrukte teks sover moontlik te vermy, terwyl soveel moontlike afbrekingspunte wel verskaf word. Alhoewel 2% meer geleenthede gemis word as ALG5 gebruik word, maak dit 131 (1,3%) minder foute. Verder sien ons dat ALG2 verdelingsfoute in twee minder woorde as ALG5 maak, maar dat dit in 108 (1%) meer woorde verdelings mis, en dus die sekondêre doelwit, naamlik om soveel moontlik verdelingspunte te verskaf, in veel minder mate as ALG5 bevredig. Ons kom dus tot die gevolgtrekking dat ALG5 die optimale algoritme is.

In Tabel 8.3 toon ons weer die uitkomste op woordvlak van ALG6 en ALG5 en van ALG3 en ALG2 om die effek van die byvoeging van NNLG te illustreer. Hieruit blyk dit dat heelwat van die neurale netwerk en TEX-algoritme se foute onderling reggestel word wanneer dit gekombineer word.

Die feit dat die neurale netwerk telkens meer woorde met gemiste afbrekings veroorsaak, is nie 'n groot probleem nie. Sulke gemiste verdelings behoort nie 'n beduidende invloed op die uitleg van teks te hê nie, aangesien 'n geskikte geldige afbreekposisie naby die gemiste posisie beskikbaar behoort te wees, byvoorbeeld in *prys-ver-dienste-veel-voude*. Waar die gemiste verdeling egter

Algoritme	Reg in C_p	Reg weg C_n	Mis F_n	Fout F_p
ALG6: OPSG → OPLG	9 464	204	60	485
ALG5: OPSG → OPLG + NNLG	9 625	208	307	73
ALG3: OPSG + NNSG → OPLG	9 539	208	78	388
ALG2: OPSG + NNSG → OPLG + NNLG	9 519	208	415	71

Tabel 8.3: Illustrasie van die effek as NNLG bygevoeg word

by die grens tussen samestellende dele voorkom en lang gedeeltes nie verdeel word nie, kan dit problematies wees, soos in *ge-brui-kerswaan-sin*. Sulke gevalle behoort in 'n saamgesteldewoord-uitsonderingslys aangespreek te word, of die tegnieke moet herafgerig word om patrone uit sulke bekende foutwoorde in te sluit.

8.2 Toets van algoritme op publikasies

Om die algoritme in die praktyk te toets, is die elkeen van die tegnieke vir lettergreep- en saamgesteldewoordverdeling afgerig deur die afrigtingsdata uit die volledige datastelle van ±183 000 woorde te gebruik. Werklike voorbeelde van gedrukte materiaal is bekom, naamlik publikasies soos artikels uit koerante, tydskrifte en vaktydskrifte, asook romans en dele uit die 1983 vertaling van die Bybel.

Die woorde wat in elke stuk voorkom, is telkens deur middel van Program D.1.1 onttrek, terwyl die frekwensie van elke woord bepaal is. Die lys unieke woorde uit elke stuk is met die leksikon vergelyk en gemeenskaplike woorde se lettergreepverdeelde weergawes is onttrek (InBeide). Program D.6.1 is hiervoor gebruik. Hierdie woorde is soos voorheen met ALG5 in lettergrepe verdeel en teen die korrekte verdelings getoets (Program D.6.3).

Die woorde wat slegs in elke stuk voorkom (NetHier), is vervolgens beskou. Anderstalige woorde is verwyder en die oorblywende woorde is met ALG5 in lettergrepe verdeel. (Program D.6.4.) Die lettergreepverdelings is per hand beoordeel en die resultate is met dié van InBeide gekombineer om 'n totale beeld te kry. Woorde waarin foute gemaak is se frekwensies in die stuk is in aanmerking geneem sodat die totale aantal foute telkens gebruik is om 'n gevolgtrekking te maak.

Die toetsproses word vervolgens vir 'n koerantartikel en 'n artikel uit 'n vaktydskrif volledig bespreek.

'n Koerantartikel

Die artikel *Vooruitsig vir indiensneming tot Junie is 0%* deur Vida Booysen wat in Beeld van 13 Maart 2013 verskyn het, word in Figuur 8.1 getoon. Let op dat woordafbreking in 44 van die 120 reëls gebruik is, dus in 36,7% van die reëls.

Vooruitsig vir indiensneming tot Junie is 0%

Vida Booysen

Bloemfontein. – Werkgewers in Suid-Afrika verwag dat die arbeidsmark van April tot Junie in groot mate sal stilstaan. Tog moet werksoekers nie moed verloor nie, want daar is wel geleenthede in sektore soos klein- en groothandel, elektrisiteit en in die vervoer-, bergings- en kommunikasiebedryf.

Dít is volgens die werwingsgroep Manpower se jongste opname oor indiensnemingsvooruitsigte wat gister uitgereik is. Dié opname word elke kwartaal in 42 lande gedoen en meet werkgewers se voorneme om óf meer werkers aan te stel óf hul arbeidsmag te verklein. Sodoende word 'n netto vooruitsig vir indiensneming verkry.

Vir Suid-Afrika is die algehele netto vooruitsig vir April tot Junie 0%, wat daarop dui dat die persentasie werknemers wat meer mense in diens wil neem, gelykstaande is met die aantal wat in die volgende drie maande werkers gaan aflê. Vergeleke met die ooreenstemmende tydperk verlede jaar is die vooruitsig vir indiensneming vanjaar 'n raps beter, met 'n toename van 2%.

Werkgewers se versigtigheid is deels toe te skryf aan arbeidskwessies, sê Lyndy van den Barselaar, besturende direkteur van Manpower SA.

"Die onlangse vlaag van stakings wat van die mynbousektor tot ander sektore soos die landbou versprei het, het 'n negatiewe uitwerking gehad op baie bedrywe se indiensnemingsplanne.

"Voeg hierby die styging in die minimum loon, strenger vereistes vir swart ekonomiese bemagtiging (SEB) en die hersiening van die grondhervormingsbeleid, en jy het 'n situasie waar baie werkgewers besluit het om meer konserwatief te wees met indiensneming," sê Van den Barselaar.

Altesame 87% van die deelnemers aan die Manpower-opname verwag geen verandering aan hul personeelgetal in die tweede kwartaal nie, terwyl 5% hul personeel gaan verklein en 7% van plan is om meer mense in diens te neem. Werkgewers in die groot- en kleinhandelsektor het die meeste optimisme getoon, met 'n netto indiensnemingsvooruitsig van 7%. Meer werkgeleenthede word ook verwag in die elektrisiteits-, gas- en waterbedryf, finansies, versekering en sakedienste, die staatsdiens en in die vervoer-, bergings- en kommunikasiebedryf.

Sektore waarin afleggings verwag word, sluit in die hotel- en restaurantbedryf, vervaardiging en konstruksie.

"Die vervaardiging- en konstruksiesektor word steeds deur 'n swakker verbruikersvraag, versigtige sakebesteding en dalende uitvoer geknou," sê Van den Barselaar. Verbruikers werk versigtig met hul geld en daarom kry sektore soos hotelle en restaurante ook swaar, met die indiensnemingsvooruitsig wat met 2% afneem teenoor die eerste kwartaal.

Indiensneming in die sektore vir basiese dienste soos elektrisiteit- en watervoorsiening groei steeds, maar die verwagting is dat dit kan afneem vanweë die nasionale energiereguleerder (Nersa) se besluit om 'n laer verhoging in elektrisiteitstariewe aan Eskom toe te staan as waarvoor die kragreus gevra het.

As die indiensnemingsvooruitsigte per provinsie ontleed word, is werkgewers in Gauteng met netto indiensneming van -2% meer pessimisties as in die res van die land, terwyl die Oos-Kaap die beste vooruitsig van 5% het. Die pas van indiensneming is stadig in die Vrystaat en Wes-Kaap, waar die netto vooruitsig vir die tweede kwartaal 1% is.

"Hoewel ons gehoop het op beter resultate ná die feestyd se sterk verbruikersbesteding en positiewe beloftes op die ANC se Mangaung-konferensie, vertoon die sakesektor steeds gedemp en neem baie ondernemings 'n 'wag-en-sien-houding' aan voordat hulle hulle verbind tot enige grootskaalse groeiplanne," het Van den Barselaar gesê.

Figuur 8.1: Beeld 13-03-2013

Die artikel bevat 'n totaal van 528 woorde, waarvan 264 unieke woorde is. In Tabel 8.4 word die frekwensies van woorde wat dikwels voorkom, getoon.

die	42	aan	6	het	9	sektore	5
en	23	indiensneming	6	met	9	vooruitsig	5
in	19	meer	6	se	7	barselaar	4
van	17	te	6	vir	7	den	4
is	12	wat	6	word	6	hul	4
'n	9	werkgewers	6	netto	5	kwartaal	4

Tabel 8.4: Woorde wat meer as een keer in artikel voorkom

Die unieke woorde is met die algoritme in lettergrepe verdeel en die resultate is soos volg:

1. Geen koppeltekens is op verkeerde plekke in enige van die woorde geplaas nie. Daar is dus geen risiko vir foute in die artikel nie.
2. Lettergreepverdelings is in nege woorde gemis. Voorbeelde hiervan is *afri-ka*, *barse-laar*, *dienste*, ens. Hierdie gevalle sluit eiename soos *mangaung* en *nersa* in wat as werklike uitsonderings beskou kan word. Gemiste afbrekingspunte het tot gevolg dat afbreking met ALG5 nie op presies dieselfde plek as in die gedrukte weergawe sou wees nie – in hierdie geval by *Bar-selaar* soos dit in die tweede kolom, paragraaf twee voorkom. Die volgende afbreekpunt wat twee letters verder is, sou waarskynlik aanvaarbaar wees vir die uitleg van die stuk.

Dit is duidelik dat die algoritme in hierdie geval baie goed presteer. In Tabel 8.5 word die algoritme se verdeling van die langste woorde in die stuk getoon. (Let op die gemiste verdeling.)

in-diens-ne-mings-voor-uit-sig-te	in-diens-ne-mings-plan-ne
in-diens-ne-mings-voor-uit-sig	ver-brui-kers-be-ste-ding
elek-tri-si-teits-ta-rie-we	e-ner-gie-re-gu-leer-der
grond-her-vor-mings-be-leid	kom-mu-ni-ka-sie-be-dryf

Tabel 8.5: Lettergreepverdeling van langste woorde in artikel

'n Artikel uit 'n vaktydskrif

'n Artikel deur Lesley le Grange [LG12] wat in die *Suid-Afrikaanse Tydskrif vir Natuurwetenskap en Tegnologie* (SATNT) gepubliseer is, is ontleed. Dit is duidelik dat geen woordafbreking in hierdie tydskrif toegelaat is nie, terwyl dubbelgeskouerde teks gebruik is. Die gevolg hiervan is dat die teks in sommige dele onegalig is as gevolg van die teenwoordigheid van lang woorde. In Figuur 8.2 word 'n voorbeeld hiervan getoon waar die voorkoms van "eilande" en "riviere" van wit spasie duidelik waarneembaar is.

Hierdie artikel is ontleed en die volgende is waargeneem:

- ▷ Die artikel bevat 'n totaal van 5 444 woorde waarvan 1 434 unieke woorde is.

Sedert 1994 is verskeie nasionale kurrikulumraamwerke in Suid-Afrika beskikbaar gestel. Die oorgang vanaf 'n inhoudgebaseerde Tussentydse Kernleerplan na 'n Uitkomsgebaseerde Nasionale Kurrikulumverklaring verteenwoordig vermoedelik 'n belangrike skuif in kurrikulumbenadering. Alhoewel die implementering van uitkomsgebaseerde onderrig of leer 'n impak op onderwysers se werk met inbegrip van lesbeplanning en klaskamerorganisasie uitgeoefen het, het die onderliggende kurrikulum (kurrikulumparadigma) dieselfde gebly. Die

Figuur 8.2: 'n Onegalige paragraaf uit SATNT

- In 'n vergelyking van die woorde met die leksikon, kom 1 163 woorde in beide voor, terwyl 269 woorde net in die artikel voorkom. Nadat die Engelse woorde, afkortings en woorde met spelfoute verwyder is, was daar 127 Afrikaanse woorde wat nie in die leksikon voorkom nie. Hierdie woorde is hoofsaaklik nuwe samestellings, soos *probleemoplossingsvaardighede*, en eiename soos *Reddy* en *Taylor* wat ingelaat is ter wille van volledigheid.

- Die woorde wat in beide lyste voorkom, is met die algoritme verdeel, getoets en die resultate is ontleed. Die algoritme het lettergrepe in 20 woorde gemis, soos *afri-kaans*, *reflek-teer*, *volgor-de*, ens. Dit het ook twee afbrekingsfoute gemaak, naamlik *be-ste* en *uit-kom-ste*. (As die vae reël rondom die letter *s* in aanmerking geneem word, kan laasgenoemde eintlik as korrek aanvaar word.)

- Vyf verdelings in die lys van woorde wat nie in ons leksikon voorkom nie, is gemis en foute is in vier woorde gemaak.

- Indien die frekwensie van woorde in aanmerking geneem word, het die algoritme in 12 uit die 5 444 woorde foute gemaak. Die waarskynlikheid dat 'n afbrekingsfout in die stuk sou voorkom – vir enige bladuitleg– is dus 0,0022 (0,22%).

- Indien ons in aanmerking neem dat daar in die gedrukte artikel ongeveer agt woorde per reël voorkom, neem die waarskynlikheid vir 'n afbrekingsfout in die artikel af tot ±0,03%.

Indien woordafbreking wel in die tydskrif toegelaat sou word, sou die onegalige paragraaf hierbo soos in Figuur 8.3 gelyk het.

Sedert 1994 is verskeie nasionale kurrikulumraamwerke in Suid-Afrika beskikbaar gestel. Die oorgang vanaf 'n inhoudgebaseerde Tussentydse Kernleerplan na 'n Uitkomsgebaseerde Nasionale Kurrikulumverklaring verteenwoordig vermoedelik 'n belangrike skuif in kurrikulumbenadering. Alhoewel die implementering van uitkomsgebaseerde onderrig of leer 'n impak op onderwysers se werk met inbegrip van lesbeplanning en klaskamerorganisasie uitgeoefen het, het die onderliggende kurrikulum (kurrikulumparadigma) dieselfde gebly.

Figuur 8.3: Paragraaf uit SATNT met afbreking toegelaat

8.2.1 Resultate met meerdere voorbeelde uit die praktyk

Ten einde 'n globale beeld van die algoritme se prestasie in die praktyk te kry, is die volgende publikasies uit 'n verskeidenheid van bronne beskou:

- Koerantartikels
  - Beeld (2013-04-28): *Madiba-miljoene vir gesin* deur Andrew Trench, Thanduxolo Jika en Jeanne van der Merwe;
  - Rapport (2013-04-20): *Hanlie Retief gesels met Rachel Jafta* deur Hanlie Retief;
  - Finansies en Tegniek (2011-07-25): *Maak geld (al verdien jy dit nie)* deur Vic de Klerk;
  - Finansies en Tegniek (2011-07-21): *Dis 'n borrel* deur Marc Ashton en Simon Dingle;
- Artikels uit 'n vaktydskrif
  - SATNT (2012-11-14): *Veranderinge in skoolbiologie in Suid-Afrika ná apartheid* deur Lesley le Grange;
  - SATNT (2013-04-17): *'n Teoretiese besinning oor die implikasies van die filosofie van tegnologie vir klaskamerpraktyk* deur Piet Ankiewicz;
  - SATNT (Desember 2008): *Elementêre topologie en berekeningsuniversaliteit* deur Petrus Potgieter;
- Natuurwetenskaplike artikels
  - Litnet (2012-06-08): *Antimikrobiese nanovesels vir waterbehandeling: poli(viniel-alkohol)- en poli(akrielonitriel)-nanovesels met silwer-nanopartikels* deur Danielle du Plessis, *et al*;
  - Litnet (2012-08-15): *Epigenetika: die skakel tussen genetika en omgewing* deur V O´Neill, *et al*;
- Kortverhale
  - Rooi Rose (Maart 2013): *Blikkiesmelkbeskuit* deur Albi Prinsloo;
  - Rooi Rose (2013-02-11): *Ou liefde roes nie* deur Helena Opperman;
  - Verhale deur Koos Kombuis, Amanda Strydom, Anna-Mart van der Merwe en Neels van Jaarsveld uit *SARIE 40 Flitsverhale*, saamgestel deur André le Roux;
- Romans
  - *Brug van die Esels* deur Dalene Matthee;
  - *Griet kom weer* deur Marita van der Vyver;
  - *Laat vrugte* deur C M van den Heever;
- Die Bybel (1983 vertaling)
  - *Johannes*;
  - *Genesis*.

In Tabel 8.6 word die resultate van die algoritme op hierdie publikasies getoon. Ons sluit vir elke stuk die totale aantal woorde, die aantal unieke woorde (dus sonder herhaling), die aantal woorde wat met die leksikon ooreenstem en die aantal woorde wat net in die stuk voorkom (met vreemde woorde verwyder), in. Verder toon ons die totale aantal misse, die unieke Afrikaanse woorde met foute (F), ander foute soos in vreemde eiename, verafrikaanste Engelse woorde, akronieme, ensovoorts (A) en die aantal werklike foute met frekwensie in ag geneem (Fr). Die risiko dat die algoritme 'n fout in enige woord sal maak, word in die laaste kolom as 'n persentasie gegee. Dit is nie die risiko gebaseer op reëllengte nie, wat in alle gevalle heelwat laer is.

Ons sluit ook die ontleding van 'n groot korpus wat onder andere ook die voorbeelde hierbo insluit, in. In die laaste geval het ons ook die woorde in hierdie proefskrif ontleed.

8.2.2 Bespreking

Alhoewel die algoritme duidelik goed presteer, maak dit foute in verskeie tipes woorde. Een so 'n geval is by eiename van vreemde oorsprong. Engelse name wat die klinkerkombinasie *ae* bevat wat as een klank uitgespreek word, is gewoonlik 'n probleem, byvoorbeeld *re-a-gan*. In Afrikaans het ons die woord *reageer* wat as *re-a-geer* verdeel word, wat die fout in *reagan* verklaar. Netso bevat eiename dikwels medeklinkerkombinasies wat as een klank uitgespreek word, maar nie in Afrikaans voorkom nie, en foute in name soos *nd-lovu*, *tuk-wini*, *m-clachlan*, ensovoorts veroorsaak.

Verafrikaanste Engelse woorde word dikwels in moderne skryfwerk gebruik, en het afbrekingsfoute tot gevolg. Voorbeelde wat in die publikasies onder beskouing voorgekom het, is *uitgepuzzle*, *gescheme*, *crossover-troue*, ens.

Soos verwag, kom foute dikwels rondom die letter *s* voor, byvoorbeeld *doods-on-de*, *wan-hoop-spo-ging*, *afbreking-steenstrydigheid*, ensovoorts. Verder word saamgestelde woorde waarvan die tweede deel met 'n klinker begin, dikwels in daardie posisie verkeerd verdeel, byvoorbeeld *a-part-hei-de-ra*, *we-ten-ska-pop-voe-ding* en *kur-ri-ku-lu-maan-pas-sing*.

Verdelings word dikwels gemis waar vreemde letterpatrone as gevolg van die samestelling van woorde ontstaan. As OPSG so 'n verdeling mis en die resultate van OPLG en NNLG verskil, word die verdeling heeltemal gemis. 'n Ontleding van die neurale-netwerkwaardes en die waardes in die TEX-patrone kan dalk 'n oplossing bied vir sulke gevalle, maar dit word vir verdere navorsing gelaat.

Woorde wat in bogenoemde bronne voorkom en wat foutiewe verdelings bevat, word in 'n uitsonderingslys geplaas. Ook saamgestelde woorde waarin die verdeling tussen die samestellende dele gemis word, word in die uitsonderingslys geplaas – veral as dit lang gedeeltes veroorsaak wat geen verdeling het nie, byvoorbeeld *vrou-e-tydskrif-te*, *ver-brui-kerswaan-sin*, ensovoorts.

Publikasie	Aantal woorde				Misse	Foute			Risiko
	Totaal	Uniek	In LG_Lys	Net hier		F	A	Fr	(%)
Beeld (Vida Booysen)	528	264	220	34	9	0	0	0	0,00
Beeld (Trench, *et al*)	552	277	207	70	9	2	4	2	0,36
Rapport (Hanlie Retief)	1 783	724	618	106	13	2	1	2	0,11
Fin & Tegniek (Vic de Klerk)	1 777	591	530	37	7	2	3	2	0,11
Fin & Tegniek (Ashton & Dingle)	3 014	981	793	188	14	2	4	3	0,10
SATNT (Lesley le Grange)	5 444	1 434	1 163	127	31	6	0	12	0,22
SATNT (Piet Ankiewicz)	5 971	1 407	1 092	315	32	9	3	25	0,42
SATNT (Petrus Potgieter)	2 420	537	470	63	2	3	0	6	0,25
Litnet (D du Plessis, *et al*)	3 196	907	663	239	14	4	5	5	0,16
Litnet (V ONeill, *et al*)	7 130	1 906	1 231	674	60	7	41	21	0,29
Rooi Rose (Albi Prinsloo)	1 547	543	504	35	8	1	0	2	0,10
Rooi Rose (Helena Opperman)	2 268	720	696	19	10	1	0	1	0,04
Sarie Flitsverhale	5 582	1 514	1 303	166	18	3	3	18	0,32
Brug van die Esels	98 524	6 691	6 331	358	96	10	4	54	0,05
Griet kom weer	99 480	9 545	8 749	796	145	19	11	48	0,05
Laat vrugte	94 279	8 014	6 859	1 154	92	31	6	104	0,10
Johannes	20 679	1 436	1 389	38	14	3	0	4	0,02
Genesis	37 157	2 742	2 737	367	36	49	3	11	0,13
Groot korpus	397 670	23 538	18 706	4 832	413	84	81	281	0,07
Hierdie proefskrif	28 270	3 352	2 352	941	86	19	6	71	0,25

Tabel 8.6: Resultate van algoritme toegepas op 'n verskeidenheid publikasies

Hoofstuk 9

Bespreking en verdere navorsing

9.1 Bespreking

Die doel van hierdie studie was om te bepaal tot watter mate die woordafbrekingsprobleem (of lettergreepverdelingsprobleem) in Afrikaans deur 'n suiwer meganiese, patroongebaseerde benadering oplosbaar is. Geen konteks, sintaksis of semantiese inligting word dus in aanmerking geneem nie. Ons het die masjienleertegnieke neurale netwerke, beslissingsbome en die TEX-algoritme vir die lettergreepverdelingsprobleem afgerig en ondersoek. Ons het bevind dat laasgenoemde die beste resultate lewer met 99,56% akkuraatheid op verdelingsgeleentheidsvlak. Dit is die beste prestasie wat ons kon bereik na intensiewe ontleding van elke masjienleertegniek op sy eie.

In 'n poging om hierdie prestasie verder te verbeter, het ons die K-algoritme uit 'n kombinasie van masjienleertegnieke ontwerp. Alhoewel dit marginaal laer akkuraatheid op verdelingsgeleentheidsvlak, naamlik 99,54% lewer, maak dit 1,3% minder lettergreepverdelingsfoute as die TEX-algoritme op sy eie.

In die proses om die masjienleertegnieke af te rig en te toets, het ons twee verdeelde weergawes van ons leksikon van ±183 000 woorde saamgestel – een waarin woorde in lettergrepe verdeel is en een waarin saamgestelde woorde in hul samestellende dele verdeel is. Na ons mening is hierdie woordelyste grootliks korrek. Die prestasie van die K-algoritme kan verbeter word deur bykomende afrigtingsdata vir die neurale netwerk en die TEX-algoritme in te samel uit omgewings soos mediese verslae en publikasies, regsdokumente, wetenskaplike en tegniese artikels, gepubliseerde fiksie, ensovoorts. Dit is natuurlik uiters belangrik om die lettergreep- en saamgesteldewoordverdeling van alle nuwe woorde streng te kontroleer, aangesien masjienleertegnieke se prestasie afhanklik is van korrekte afrigtingsdata.

'n Rekursiewe algoritme (die SG-algoritme) is ontwikkel om saamgesteldewoordverdeling te doen. Alhoewel die prestasie van die TEX-algoritme op saamgesteldewoordverdeling goed is (92,63% op woordvlak), maak dit steeds heelwat verdelingsfoute wat die K-algoritme se prestasie negatief beïnvloed. Om hierdie foute uit te skakel, kan die moontlikheid ondersoek

word om die SG-algoritme saam met die TEX-algoritme vir die taak aan te wend. Die tekortkoming van die SG-algoritme dat verdelings nie gedoen word wanneer die spelling van woorde tydens verbuiging verander en die verbuiging nie in die verwysingslys voorkom nie, sal aangespreek moet word om te verseker dat verdelings in sulke gevalle nie gemis word nie.

In die K-algoritme word posisies waar die TEX-algoritme en die neurale netwerk se resultate verskil, geïgnoreer. Dit veroorsaak dat geldige afbrekingsposisies wat moontlik in een van die twee uitvoere voorgekom het, gemis word. Om hierdie probleem moontlik aan te spreek, kan ons die neurale netwerk se berekende waardes, asook die waardes wat in die TEX-patrone voorkom by sulke posisies ontleed om te bepaal of daar 'n reëlmaat is. 'n Meganisme kan dan ingebou word om gemiste verdelings onder sekere omstandighede uit te skakel. Ook waar die neurale netwerk en die TEX-algoritme dieselfde foutiewe uitvoer gee en 'n foutiewe verdeling dus in die uitvoer voorkom, kan hierdie waardes moontlik 'n aanduiding gee oor die risiko vir 'n fout, wat gebruik kan word om foute te vermy.

Die K-algoritme is na ons mening reeds bruikbaar in die praktyk – veral in kombinasie met 'n uitsonderingslêer. Die verbeteringstegnieke wat hierbo bepreek is, kan moontlik die algoritme se prestasie marginaal verbeter, maar daar is aspekte wat nie met patroongebaseerde metodes aangespreek kan word nie. Dit sal byvoorbeeld nooit in staat wees om tussen woorde met dieselfde spelling wat, afhangende van die konteks, verskillend verdeel word, te onderskei nie, soos *gek-lik* teenoor *ge-klik*. Om probleme soos hierdie aan te spreek, sal morfologiese, sintaktiese en/of konteksinligting gebruik moet word, wat buite die bestek van hierdie studie val. Selfs met sulke inligting beskikbaar, is daar egter gevalle wat nie rekenaarmatig oplosbaar is nie, soos in die geval van *san-daal* en *sand-aal* wat op bladsy 15 bespreek is.

Om die K-algoritme in die praktyk te kan gebruik, sal dit nodig wees om 'n toepassing vir algemeen gebruikte platvorms soos *MSWord* en *OpenOffice* te ontwikkel. Dit sal natuurlik belangrik wees dat die algoritme nie te veel geheue gebruik nie en vinnig resultate lewer.

So 'n toepassing sal uitsonderingslêers vir beide lettergreep- en saamgesteldewoordverdeling insluit. Verder sal 'n meganisme in plek gestel moet word waardeur gebruikers terugvoer kan gee sodat hierdie lêers opgedateer kan word en die verrigting van die algoritme verbeter kan word.

Ons het aangetoon dat die Afrikaanse woordafbrekers wat tans beskikbaar is vir gebruik in *MSOffice* nie onfeilbaar is nie. Daar is dus ruimte vir verbetering op hierdie hulpmiddels wat moontlik deur hierdie benadering geskep kan word.

Die lettergreepverdelingspatrone wat vir TEX gegenereer is, is aan die TEX-gemeenskap geskenk en sal in die toekoms by verstek met nuwe installasies beskikbaar wees.

Bylae A

Matlab-probleme en oplossings

A.1 Datagenereringsfoute

Ons het opgelet dat die aantal datapare wat deur Matlab gebruik word, nie altyd met die invoerdata ooreengestem het nie en ook nie die gewensde resultate gelewer het nie. Hierdie probleem het voorgekom omdat daar "gate" (ontbrekende data) in die invoerdata voorgekom het as gevolg van onbekende karakters wat in die woorde waaruit die afrigtingsdata gegenereer is, voorgekom het. Die karakters * of ! het byvoorbeeld voorgekom en geen voorsiening was in die kodering daarvoor gemaak nie.

Die funksie `FIXUNKNOWNS` wat by verstek deur Matlab gebruik word, skep vir elke data-item met onbekende data twee invoerreëls – een waarin die onbekende waardes deur die gemiddelde waarde vervang word en 'n tweede een wat aandui of die waarde vervang is (1), of nie (0).

Data-items met ontbrekende data kan met die volgende Matlab-opdragte geïdentifiseer word:

```
tf=isnan(Invoer);
[r,c]=ind2sub(size(tf),find(tf));
[r,c]
```

Sulke data kan uit die datastel verwyder word deur die volgende opdragte:

```
incompleteRows = any(isnan(Invoer), 2);
Invoer = Invoer(~incompleteRows,:);
Teiken = Teiken(~incompleteRows);
```

'n Beginselfout is egter hier uitgewys en die probleem is by die bron (die oorspronklike datastel waaruit die afrigdata gegenereer word) reggestel deur alle onbekende karakters uit die datastel te verwyder. Die funksie `FIXUNKNOWNS` is uitgeskakel deur dit in die program met {} te vervang.

A.2 Simulasiefoute

Tydens simulasie het Matlab onverwagte waardes gelewer, naamlik vir die LOGSIG oordragfunksie wat waardes tussen 0 en 1 behoort te lewer, het dit waardes buite hierdie interval gelewer. Die rede hiervoor was dat die `MAPMINMAX`-funksie die invoer by verstek op die interval $[-1;\ 1]$ afbeeld. Normaalweg veroorsaak dit nie probleme met klassifikasie nie, maar aangesien die spesifieke uitvoerwaardes belangrik is by die bepaling van lettergreepverdelingspunte, moes ook hierdie funksie afgeskakel word deur dit in die program met {} te vervang.

A.3 Vroeë terminasie van afrigting

Wanneer gepoog word om 'n neurale netwerk met baie verborge neurone af te rig, mag dit gebeur dit dat afrigting aanvanklik baie stadig plaasvind en die afrigtingsgradiënt tot onder die verstekwaarde van 1e-6 afneem. In sulke gevalle word afrigting gestaak en die doelwit word nooit breik nie.

Soms kan die probleem opgelos word deur afrigting weer te inisieer. Nuwe aanvanklike gewigte mag 'n groter aanvangsgradiënt lewer sodat afrigting kan voortgaan. Dit los egter nie altyd die probleem op nie.

'n Ander manier om hierdie probleem te oorkom, is om die verstekwaarde van die minimum gradiënt te verander. Ons het die minimum na 1e-10 verlaag en het daarna nie weer probleme met vroeë terminasie gehad nie.

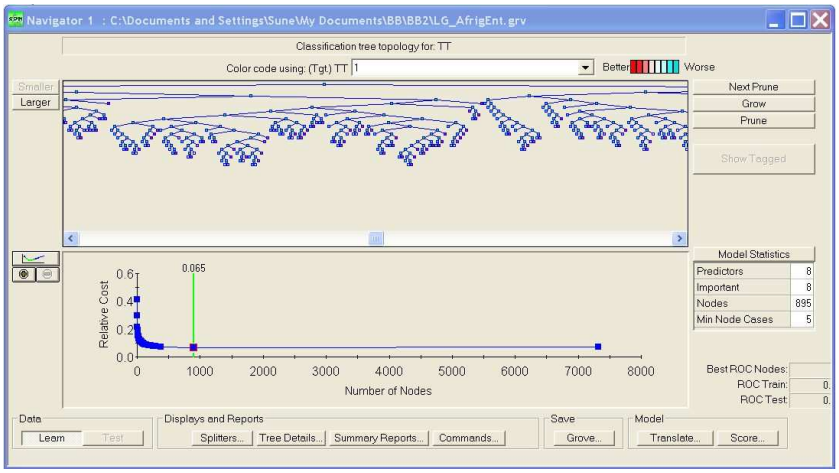
Bylae B

Die finale BB in terme van CART se grafiese uitvoer

Baie van die inligting wat in hierdie bylae verskaf word, kom uit die CART 6.0 gebruikersgids [Sys13].

Die grafiese platvorm van die CART-pakket wat ons gebruik het, is gebruikersvriendelik en verskaf volledige inligting oor 'n afgerigte boom. Hierdie inligting word in 'n *grove*-lêer vervat.

Figuur B.1 toon die *grove*-lêer vir die finale Beslissingsboom (BB) wat vir lettergreepverdeling afgerig is. Hierdie BB is met die volledige stel afrigtingsdata (unieke afrigpare) afgerig en *Entropie* is as verdelingskriterium gebruik.



Figuur B.1: Die *grove*-lêer vir die finale boom vir woordafbreking

Die BB in die boonste venster is te groot om volledig te vertoon. Dit is egter moontlik om dit rond te skuif en besonderhede van nodusse te besigtig. Vir so 'n groot boom is dit nie sinvol om na die volledige boom te kyk nie, maar opsommende statistiek is beskikbaar en kan gesien word deur op *Summary Reports* te klik.

Die relatiewe-kostegrafiek in die onderste venster van die *Grove*-diagram toon dat die boom

met minimum koste van 0,065 die optimale boom is. Hierdie relatiewe koste wat tydens kruisvalidasie bereken word, is die verhouding tussen klassifikasiefoute en die grootte van die boom.

Uit die modelstatistiek (regs) sien ons dat die optimale boom 895 eind- of blaarnodusse bevat, dat al agt veranderlikes (*Predictors*) belangrik is en dat 'n minimum van vyf gevalle in blaarnodusse voorkom.

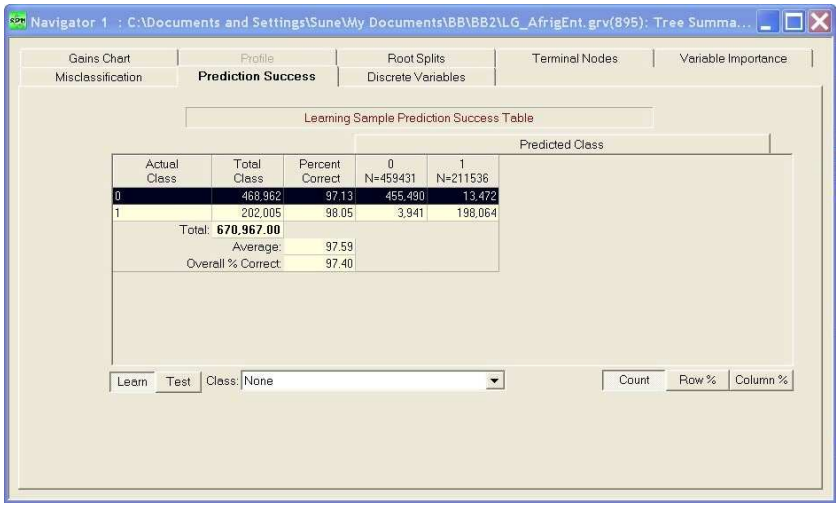
Vanuit die *Grove*-lêer kan opsommende verslae oor die prestasie van die BB verkry word. Die verskillende verslae word vervolgens bespreek.

Voorspellingsukses

Die voorspellingsukses-opsomming in Figuur B.2 verskaf statistiek oor die aantal en persentasie van gevalle wat tydens afrigting (en kruisvalidasie) reg geklassifiseer is as "plaas 'n koppelteken in" (Klas 1), of "geen koppelteken nie" (Klas 0). Let daarop dat teikenwaardes soos in die afrigtingsdatastel in die *Actual Class*-kolom gegee word, terwyl die boom se prestasie in die kolomme aangedui deur 0 en 1 verskaf word.

Die BB se voorspellingsukses word soos volg gerapporteer: Dit klassifiseer 455 490 van die 468 962 Klas 0-gevalle korrek, dus 97,13%, terwyl dit 198 064 van die 202 005 Klas 1-gevalle korrek klassifiseer, dus 98,05%. Die gemiddelde is bloot die gemiddelde van hierdie twee persentasies, terwyl die Algehele % korrek (*Overall % Correct*) die som van die korrek geklassifiseerde gevalle gedeel deur die totale aantal gevalle is, met ander woorde

$$(455\,490 + 198\,064)/670\,967 = 0{,}97405 = 97{,}40\%.$$



Figuur B.2: Voorspellingsukses

Misklassifikasie

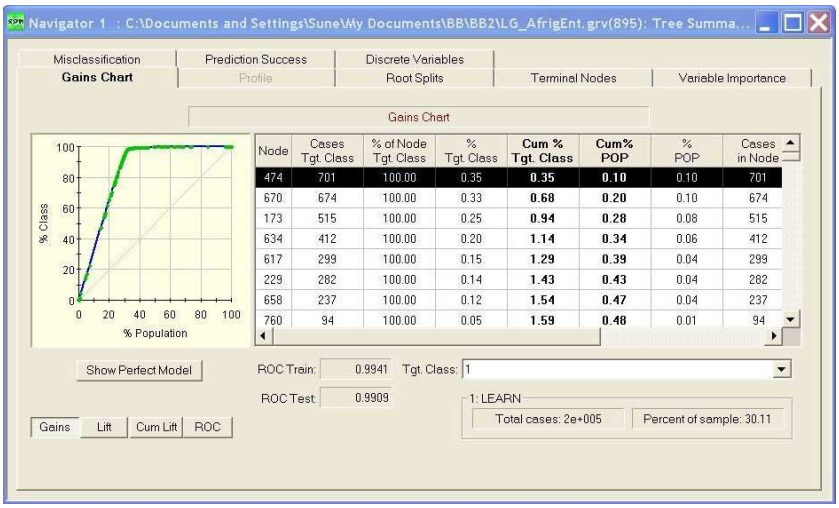
Die misklassifikasieverslag in Figuur B.3 toon die aantal gevalle wat verkeerd geklassifiseer is tydens afrigting en kruisvalidasie. Let daarop dat die volle datastel vir beide afrigting (*Learning*

Sample) en kruisvalidasie[1] (*Test Sample*) gebruik is.

Figuur B.3: Misklassifikasie

Kumulatiewe akkuraatheidsprofiel

Die *Gains chart* – die kumulatiewe akkuraatheidsprofiel of kredietrisiko – word in Figuur B.4 getoon. Uit die grafiek sien ons byvoorbeeld dat byna 60% van alle Klas 1-gevalle betrek word wanneer ons die top 20% van data wat waarskynlik as Klas 1 geklassifiseer sal word, beskou. Dieselfde kan ook vir Klas 0-gevalle gedoen word deur die teikenklas (*Tgt. Class*) te verander.

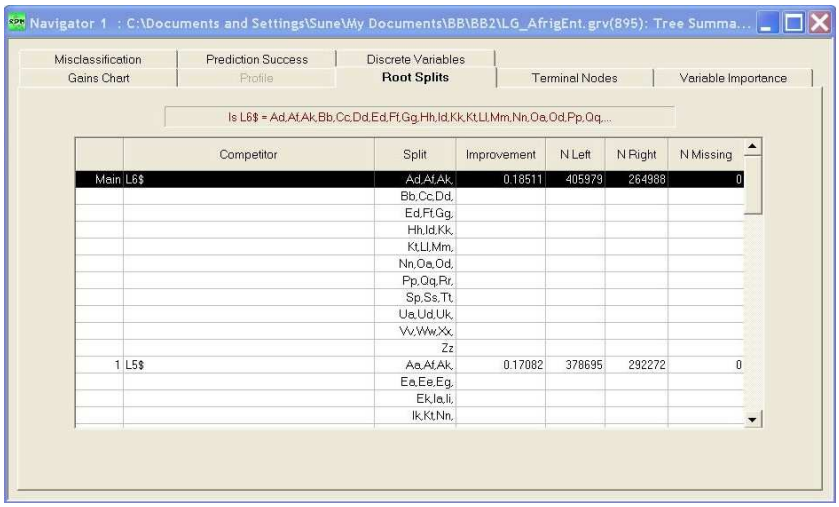


Figuur B.4: Die *Gains chart* of kumulatiewe akkuraatheidsprofiel

[1]Kruisvalidasie is 'n tegniek om modelle te toets sonder om die data in aparte afrigtings- en toetsdatastelle te verdeel.

Wortelverdelings

In die wortelnodus is al die inligting rakende data beskikbaar. Figuur B.5 toon die veranderlike(s) wat by die eerste verdeling betrokke is. Die verdeling op $L6$ lei tot 'n verbetering van 18,49% in inligtingswins. Die data word verdeel sodat 405 979 gevalle van die invoerdata na links en 264 988 na regs gestuur word.



Figuur B.5: Wortelverdelings

Belangrikheid van veranderlikes

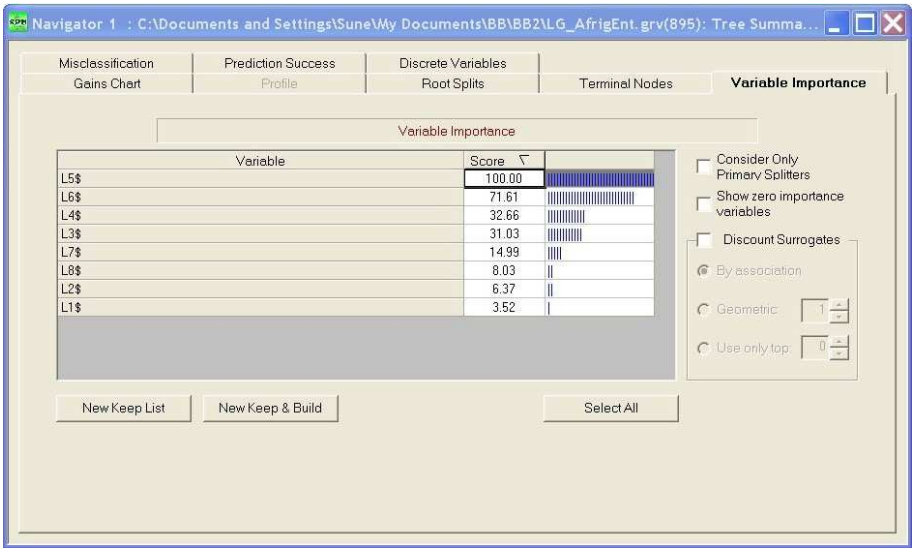
Veranderlikes word as belangrik beskou (i) as dit die primêre verdeler is wat werklik die data in die nodus verdeel, en (ii) as dit as 'n surrogaatverdeler optree wanneer die primêre verdeler nie teenwoordig is nie.

Volgens die verslag in Figuur B.6 is die veranderlikes in die eerste twee posisies ná die verdelingsposisie ($L5$ en $L6$) die belangrikste vir lettergreepverdeling.

Verdelingsreëls

Die reëls wat die beslissingsboom se verdelings vir elke blaarnodus beskryf, word verkry deur op *File/View Rules* te klik. Hierdie reëls is aangepas om in 'n Perl-program te gebruik. Elke reël begin by die wortelnodus en loop deur die boom tot by 'n blaar- of eindnodus. Die volgende is 'n voorbeeld van die reëls:

```
#/*Terminal Node 1*/
   if ((L5$ eq "Af" | L5$ eq "Ak" | L5$ eq "Hh" | L5$ eq "Ia" | L5$ eq "Ik" | L5$ eq "Kt" | L5$ eq "Ll" | L5$ eq "Mm" | L5$ eq "Nn" | L5$ eq "Oa" |
        L5$ eq "Og" | L5$ eq "Ok" | L5$ eq "Qq" | L5$ eq "Rr" | L5$ eq "Ua" | L5$ eq "Uk" | L5$ eq "Xx" | L5$ eq "Ya" | L5$ eq "Zz" ) &
       (L6$ eq "Ad" | L6$ eq "Af" | L6$ eq "Ak" | L6$ eq "Bb" | L6$ eq "Cc" | L6$ eq "Dd" | L6$ eq "Ed" | L6$ eq "Ff" | L6$ eq "Gg" | L6$ eq "Hh" |
        L6$ eq "Id" | L6$ eq "Kk" | L6$ eq "Kt" | L6$ eq "Ll" | L6$ eq "Mm" | L6$ eq "Nn" | L6$ eq "Oa" | L6$ eq "Od" | L6$ eq "Pp" | L6$ eq "Qq" |
        L6$ eq "Rr" | L6$ eq "Sp" | L6$ eq "Ss" | L6$ eq "Tt" | L6$ eq "Ua" | L6$ eq "Ud" | L6$ eq "Uk" | L6$ eq "Vv" | L6$ eq "Ww" | L6$ eq "Xx" |
        L6$ eq "Zz"))
 {
   class = 0;
 }
#/*Terminal Node 2*/
elsif (($L4 eq "Ad" | $L4 eq "Ak" | $L4 eq "Bb" | $L4 eq "Cc" | $L4 eq "Dd" | $L4 eq "Ed" | $L4 eq "Eg" | $L4 eq "Gg" | $L4 eq "Hh" | $L4 eq "Ia" |
        $L4 eq "Id" | $L4 eq "Jj" | $L4 eq "Kk" | $L4 eq "Ll" | $L4 eq "Mm" | $L4 eq "Nn" | $L4 eq "Oa" | $L4 eq "Og" | $L4 eq "Pp" | $L4 eq "Qq" |
        $L4 eq "Rr" | $L4 eq "Tt" | $L4 eq "Ud" | $L4 eq "Vv" | $L4 eq "Ww" | $L4 eq "Ya" | $L4 eq "Zz" ) &
       ($L3 eq "Ad" | $L3 eq "Af" | $L3 eq "Ak" | $L3 eq "Bb" | $L3 eq "Cc" | $L3 eq "Dd" | $L3 eq "Ea" | $L3 eq "Ek" | $L3 eq "Ff" | $L3 eq "Gg" |
```



Figuur B.6: Belangrikheid van veranderlikes

```
       $L3 eq "Hh" | $L3 eq "Ia" | $L3 eq "Kk" | $L3 eq "Kt" | $L3 eq "Oa" | $L3 eq "Od" | $L3 eq "Ok" | $L3 eq "Pp" | $L3 eq "Qq" | $L3 eq "Sp" |
       $L3 eq "Ss" | $L3 eq "Tt" | $L3 eq "Ua" | $L3 eq "Ud" | $L3 eq "Uk" | $L3 eq "Vv" | $L3 eq "Xx" ) &
      ($L5 eq "Aa" | $L5 eq "Ea" | $L5 eq "Ee" | $L5 eq "Eg" | $L5 eq "Ek" | $L5 eq "Ff" | $L5 eq "Gg" | $L5 eq "Ii" | $L5 eq "Jj" | $L5 eq "Oo" |
       $L5 eq "Tt" | $L5 eq "Uu" | $L5 eq "Yy" ) &
      ($L6 eq "Ad" | $L6 eq "Af" | $L6 eq "Ak" | $L6 eq "Bb" | $L6 eq "Cc" | $L6 eq "Dd" | $L6 eq "Ed" | $L6 eq "Ff" | $L6 eq "Gg" | $L6 eq "Hh" |
       $L6 eq "Id" | $L6 eq "Kk" | $L6 eq "Kt" | $L6 eq "Ll" | $L6 eq "Mm" | $L6 eq "Nn" | $L6 eq "Oa" | $L6 eq "Od" | $L6 eq "Pp" | $L6 eq "Qq" |
       $L6 eq "Rr" | $L6 eq "Sp" | $L6 eq "Ss" | $L6 eq "Tt" | $L6 eq "Ua" | $L6 eq "Ud" | $L6 eq "Uk" | $L6 eq "Vv" | $L6 eq "Ww" | $L6 eq "Xx" |
       $L6 eq "Zz"))
{
    $class = 0;
}
#/*Terminal Node 2*/
elsif ((
    .
    .
    .
#/*Terminal Node 895*/
elsif (($L7 eq "Aa" | $L7 eq "Af" | $L7 eq "Bb" | $L7 eq "Cc" | $L7 eq "Ea" | $L7 eq "Ed" | $L7 eq "Ee" | $L7 eq "Eg" | $L7 eq "Ek" | $L7 eq "Gg" |
       $L7 eq "Hh" | $L7 eq "Id" | $L7 eq "Jj" | $L7 eq "Kt" | $L7 eq "Ll" | $L7 eq "Od" | $L7 eq "Pp" | $L7 eq "Qq" | $L7 eq "Rr" | $L7 eq "Sp" |
       $L7 eq "Ud" | $L7 eq "Uu" | $L7 eq "Vv" | $L7 eq "Zz" ) &
      ($L6 eq "Ea" | $L6 eq "Ee" | $L6 eq "Eg" | $L6 eq "Ek" | $L6 eq "Ia" | $L6 eq "Ii" | $L6 eq "Ok" | $L6 eq "Ya" | $L6 eq "Yy") &
      ($L4 eq "Aa" | $L4 eq "Ad" | $L4 eq "Af" | $L4 eq "Ea" | $L4 eq "Ed" | $L4 eq "Ee" | $L4 eq "Eg" | $L4 eq "Ek" | $L4 eq "Ff" | $L4 eq "Id" |
       $L4 eq "Ii" | $L4 eq "Jj" | $L4 eq "Kt" | $L4 eq "Ll" | $L4 eq "Mm" | $L4 eq "Nn" | $L4 eq "Od" | $L4 eq "Ok" | $L4 eq "Oo" | $L4 eq "Rr" |
       $L4 eq "Ua" | $L4 eq "Ud" | $L4 eq "Uk" | $L4 eq "Uu" | $L4 eq "Xx" | $L4 eq "Yy" | $L4 eq "Zz" ) &
      ($L5 eq "Bb" | $L5 eq "Cc" | $L5 eq "Dd" | $L5 eq "Ed" | $L5 eq "Ff" | $L5 eq "Gg" | $L5 eq "Hh" | $L5 eq "Id" | $L5 eq "Jj" | $L5 eq "Kk" |
       $L5 eq "Ll" | $L5 eq "Mm" | $L5 eq "Nn" | $L5 eq "Pp" | $L5 eq "Qq" | $L5 eq "Rr" | $L5 eq "Ss" | $L5 eq "Tt" | $L5 eq "Vv" | $L5 eq "Ww" |
       $L5 eq "Zz"  ))
{
    $class = 1;
}
```

Bylae C

Datastrukture

Hierdie beskrywing van die datastrukture wat vir die TEX-patrone gebruik word, is met geringe wysigings uit [Fic02] geneem.

Patrone word in datastrukture waaruit gegewens maklik onttrek kan word, gestoor. Liang [Lia83] het 'n datastruktuur wat hy 'n gepakte *trie* noem uit gekoppelde *trie*-strukture en indeks-*trie*s ontwikkel. Die naam *trie* kom uit die woord "re*trie*val" en word as "traai" uitgespreek. In hierdie *trie*-strukture is elke sleutel 'n reeks waardes oor 'n eindige alfabet (met m elemente) en is dus ideaal om sleutels van variërende lengte te hanteer.

C.1 Gekoppelde *trie*

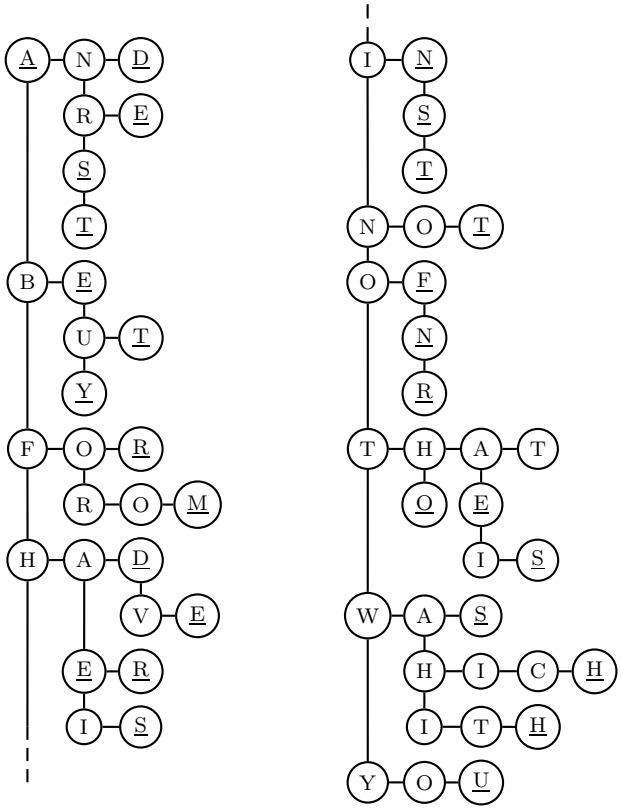
In Figuur C.1 verskyn 'n gekoppelde *trie* vir die 31 mees algemene woorde in Engels, naamlik

A	AT	FOR	HE	IN	OF	THE	WHICH
AND	BE	FROM	HER	IS	ON	THIS	WITH
ARE	BUT	HAD	HIS	IT	OR	TO	YOU
AS	BY	HAVE	I	NOT	THAT	WAS	

In so 'n struktuur word die karakters van 'n ingevoerde woord sekwensieel met die karakters in die nodusse vergelyk.

- ▷ As die letters ooreenstem, word na die volgende letter in die sleutel beweeg.
- ▷ As die letters nie ooreenstem nie, word die volgende vertakking ondersoek.
- ▷ 'n Onderstreepte letter dui die einde van 'n woord aan. As die einde van 'n woord met 'n eindpuntnodus ooreenstem, is die woord deel van die stel woorde onder beskouing en die uitvoer is *ja* (1). Indien dit nie die geval is nie, is die uitvoer *nee* (0).

'n Soektog deur so 'n gekoppelde *trie* is taamlik stadig aangesien dit nodig mag wees om elke woordkarakter tot soveel keer as die aantal letters in die alfabet met die noduskarakters te vergelyk.



Figuur C.1: 'n Gekoppelde *trie*-struktuur vir die 31 mees algemene Engelse woorde

C.2 Indeks-*trie*

In 'n indeks-*trie* word dieselfde beginsel as hierbo gebruik, maar die vertakkings word in 'n skikking met m kolomme en $n + 1$ rye uitgevoer, waar n die aantal woorde in die *trie* is. Die elemente van die skikking dui aan watter "familie" van die *trie* volgende ondersoek moet word. 'n Groep nodusse in 'n gekoppelde *trie* word as 'n familie beskou.

'n Indeks-*trie* vir die 31 mees algemene woorde in Engels verskyn in Figuur C.2. Die soektog deur so 'n indeks-*trie* is vinnig aangesien die aanvangsposisie in die *trie* maklik bepaal kan word. Dit gebruik egter baie onnodige spasie aangesien slegs 'n paar posisies inskrywings bevat, terwyl die res leeg is. In hierdie indeks-*trie* is $26 \times 32 = 832$ posisies beskikbaar, waarvan slegs 59 gebruik word.

C.3 Gepakte *trie*

Liang [Lia83] het 'n tegniek ontwikkel wat die beste eienskappe (spasie en spoed) van bogenoemde twee tegnieke kombineer. Dit word *gepakte tries* genoem aangesien die koppelings van een familie in die leë spasies in 'n indeks-*trie*, wat vir nul-koppelings van ander families gereserveer is, ingepak word.

(Voortaan word na families as *toestande* en na posisie-aanwysers as *oorgange* verwys om met

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
1	2	5				7		11	16					17	19					20			24		31	
2														3				4	0	0						
3				0																						
4					0																					
5					0																6				0	
6																				0						
7															8			9								
8																		0								
9															10											
10													0													
11	12				14			15																		
12				0																						
13					0																					
14																		0								
15																			0							
16														0					0	0						
17															18											
18																				0						
19						0								0				0								
20								21							0											
21	22				0				23						0											
22																				0						
23																			0							
24	25							26	29																	
25																				0						
26									27																	
27			28																							
28								0																		
29																				30						
30								0																		
31															32											
32																					0					

Figuur C.2: 'n Indeks-*trie* vir die 31 mees algemene Engelse woorde

die terminologie van eindige-toestandmasjiene ooreen te stem.)

In 'n gepakte *trie* is dit nodig om aan te toon of 'n oorgang op die huidige toestand betrekking het of aan 'n ander toestand, wat in dieselfde area gepak is, behoort. Dit word gedoen deur 'n karakterindeks saam met die oorgangsindeks te stoor. Slegs wanneer die karakter onder beskouing dus met die karakterindeks ooreenstem, behoort 'n oorgang aan die toestand onder beskouing. Die bykomende vereiste dat verskillende toestande nie in dieselfde basisposisie gepak mag word nie, moet dus nagekom word.

Die *trie* word met die *eerste-pas*metode gepak. Dit beteken dat toestande een-een ingepak word met elke toestand in die plek met die laagste indeks waar dit sal inpas sonder om met enige vorige gepakte oorgange of reeds gevulde basisposisies te oorvleuel. Dikwels word byna alle oop spasies in die *trie* met oorgange van ander toestande gevul.

In Figuur C.3 is die indeks-*trie* in 'n enkelry skikking gepak – die rye in die figuur vorm eintlik een lang ry. (In hierdie geval is agtervoegselkompressie, wat in die volgende afdeling bespreek word, toegepas.)

Met 'n gepakte *trie* word die 31 mees algemene Engelse woorde dus in slegs 60 posisies ingepak in vergelyking met die 832 wat nodig was by die indeks-*trie*.

	0	1	2	3	4	5	6	7	8	9
00		A 8	B11			D0	F 3	E 0	H30	I23
10	C 5			H 0	N25	O32	E 0		O12	M0
20	T33	R14	N 1	W46	T 0	Y37	R 2	S 0	T 0	O 6
30	R 0	A29	U 4	D 0	S 0	E12	Y 0	N 0	F 0	I15
40	O 4	H44	S 0	T 0	I 7	A 4	N 0	A15	O 0	E 0
50	R 0	V 2	O38	I15	H35	I36	T 5			U 0

Figuur C.3: Gepakte *trie* vir die 31 mees algemene Engelse woorde

Om die woord FROM in die gepakte *trie* te soek, gaan ons soos volg te werk:

- ▷ Die waardes 1 tot 26 word met die letters A tot Z geassosieer. Die letter **F** stem dus met posisie 6 ooreen. In posisie 6 is die inskrywing F 3 wat 'n geldige oorgang is aangesien die karakterindeks F is. Van hier af gaan ons dus na posisie 3.
- ▷ In posisie 3 is geen inskrywing nie en ons soek van hier af die eerste **R**-karakterindeks, naamlik R14 in posisie 21. Ons gaan dus na posisie 14.
- ▷ Die inskrywing in posisie 14 is N25. Dit is nie 'n geldige oorgang nie, want ons soek 'n **O**. Ons soek dus van posisie 14 af die eerste O-karakterindeks, naamlik O32 in posisie 15 wat ons na posisie 32 stuur.
- ▷ Die inskrywing in posisie 32 is U4, wat nie 'n geldige oorgang is nie. Ons soek dus van hier af die eerste M-karakterindeks, naamlik M 0 in posisie 19. Die feit dat dit onderstreep is, dui daarop dat die woord voltooi is en wel in die datastel voorkom en die uitvoer van die soektog is 1.

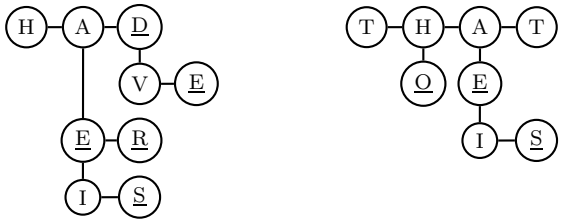
C.3.1 Minimeringsmetodes

Agtervoegselkompressie

'n Groot voordeel van *trie*-datastrukture is dat agtervoegsels wat in verskeie woorde dieselfde is, outomaties in verteenwoordigende paaie in die struktuur gekombineer kan word. Die woordeboek word dus tot 'n mate saamgepers. Om spasie te spaar, word gemeenskaplike gedeeltes van *trie*s saamgevoeg. (Ooreenkomste word van die einde van die vertakkings gesoek.)

Beskou byvoorbeeld die woorde THIS en HIS in die gekoppelde *trie* in Figuur C.1. Die toestande wat hierdie woorde bevat, word in Figuur C.4 herhaal.

Die eindnodusse van beide HIS en THIS toets vir die letter S. Hierdie twee nodusse kan gekom-



Figuur C.4: Toestande wat die woorde HIS en THIS bevat

bineer word deur hul voorgangers na dieselfde nodus te laat verwys. Die stel woorde waarna die *trie* verwys, verander nie. Die voorgangernodusse kan op dieselfde manier gekombineer word, aangesien beide vir I toets en na die S-node gaan indien suksesvol. Alhoewel die voorgangers van die I-nodusse beide E is, kan hulle nie gekombineer word nie, aangesien die een aan die linkerkant ook na R vertak.

Alle eindnodusse wat vir dieselfde karakters toets kan so saamgevoeg word. Dit alleen spaar soveel nodusse as die aantal woorde in die woordeboek plus hoogstens 26. Verder kan langer agtervoegsels soos *-ly*, *-ing*, *-tion*, ens. ook dikwels gekombineer word.

Die *trie*-gebaseerde woordeboekvoorstelling kan aangepas word om afkapping te doen deur die uitvoer te verander. In die uitvoer word dan aangedui waar die woord afgekap kan word. In plaas van 'n enkel-bisuitvoer wat sê of 'n woord in die woordeboek is of nie, word moontlike afkappingsposisies in die uitvoer aangedui.

Voorvoegselkompressie

Soos met agtervoegsels, kan voorvoegselkompressie ook gedoen word. Afkapping word byvoorbeeld aangedui sodra die letters *hyph* aan die begin van 'n woord voorkom. Die uitvoer sal sê *kap na die tweede letter af* aangesien dit vir alle woorde wat met hierdie letters begin, geld. Dit geld egter nie vir die letters *hyp* nie aangesien woorde soos *hyp-not-ic* dit weerspreek. Daar is heelwat woorde wat voorvoegels van ander woorde is, maar wat verskillend afgekap word, soos *ca-ret* en *care-ta-ker* of *as-pi-rin* en *as-pir-ing*. Hierdie probleem word aangespreek deur 'n karakter aan die einde van die woord te haak voordat dit in die gepakte *trie* geplaas word.

Bylae D

Programme

D.1 Datamanipulasie

D.1.1 GenWrdeUitKorpus.pl

'n Lys van woorde word uit 'n korpus van elektroniese teks gegenereer.

```
#!/usr/bin/perl
use utf8;

open(DATA,  "<:utf8", "Joh.txt") or die "Invoer!";              #Invoer
open(FREK,  ">:utf8", "JohFrekwensie.txt") or die "Uitvoer1!";
open(UIT,   ">:utf8", "JohUniek.txt") or die "Uitvoer2!";       #Lys van wrde in KORPUS
while ($line = <DATA>){
    $line = lc $line;
    $line =~ tr/.,!?:;""?()$&*\[\]\_/ /;                        #verwyder leestekens
    @wordsinline = split(/\s/, $line);                          #breek reel op by spasies
    push(@words, @wordsinline);
    }
foreach $word (sort({lc$a cmp lc$b}@words)){                    #sorteer 'case insensitive'
    if ($word !~ /www|@|%|'|^\-|\-$|^\'|\'$|^\‘|\‘$|            #webadresse, vreemde kar.
        \/|>|<|\ |[0-9]|\w{3,}/){                               #syfers,letter > 2 keer
        push (@woorde, "$word\n");                              #skep array van woorde
        }
    }
foreach $woord (@woorde){                                       #verwyder duplikate
    $wordcount{$woord}++;                                       #kry frekwensie
    }
foreach $woord (sort keys(%wordcount)){
    print(FREK "$woord \t\t $wordcount{$woord}\n");
    print(UIT "$woord");
    }
close(DATA);
close(FREK);
close(UIT);
```

D.1.2 SplitKtFrek.pl

Die frekwensie van $n = 2, 3, 4, \ldots$ letterkombinasies voor en na koppeltekens word bepaal.

```
#!/usr/bin/perl
use utf8;

open(INVOER,  "<:utf8", "AfrLys.txt")   or die ("Invoer!!\n");       #Woorde met lg-verd
open(WEERSKK, ">:utf8", "4WEERSK.txt")  or die ("Uitvoer1!!\n");     #weerskante van kt
open(KWEERSK, ">:utf8", "4KWEERSK.txt") or die ("Uitvoer2!!\n");     #kts weerskante
open(VOORK,   ">:utf8", "4VOORK.txt")   or die ("Uitvoer3!!\n");     #voor kt
open(NAK,     ">:utf8", "4NAK.txt")     or die ("Uitvoer4!!\n");     #na kt
open(GEENK,   ">:utf8", "4GEENK.txt")   or die ("Uitvoer5!!\n");     #geen kt
open(NIEGEDEK,">:utf8", "4NIEGEDEK.txt")or die ("Uitvoer6!!\n");     #nie gedek
open(SKEEFRK, ">:utf8", "4SKEEFRK.txt") or die ("Uitvoer7!!\n");     #kt skeef na regs
open(SKEEFLK, ">:utf8", "4SKEEFLK.txt") or die ("Uitvoer8!!\n");     #kt skeef na links
#open(BEGINW, ">:utf8", "5BEGINW.txt")  or die ("Uitvoer9!!\n");     #begin van woord
#open(EINDEW, ">:utf8", "5EINDEW.txt")  or die ("Uitvoer10!!\n");    #einde van woord
$n=4;                                                                #verander komb-lengte:
                                                                     #$n = 2, 3,..., 8
foreach $woord (<INVOER>){
    $wl=length($woord);                                              #woordlengte
    for (my $i=0; $i <= $wl-$n-1; $i++){
        $komb=substr(lc$woord, $i, $n);
        push (@kombs, "$komb");
        }
    }
foreach $komb (@kombs){ $wordcount{$komb}++; }                       #kry frekwensie
foreach $komb (sort keys(%wordcount)){
    if($komb =~ /(\p{IsWord})x$n-1.'\-'/){
        print(VOORK "$komb\t$wordcount{$komb}\n"); }
    elsif($komb =~ /'\-'.(\p{IsWord})x($n-1)/){
        print(NAK "$komb\t$wordcount{$komb}\n"); }
    elsif($komb =~ /(\p{IsWord})x(($n-1)/2).'\-'.(\p{IsWord})x($n-1)/2){
        print(WEERSKK "$komb\t$wordcount{$komb}\n"); }
    elsif($komb =~ /'\-'.(\p{IsWord})x($n-2).'\-'/){
        print(KWEERSK "$komb\t$wordcount{$komb}\n"); }
    elsif($komb =~ /(\p{IsWord})x$n/){
        print(GEENK "$komb\t$wordcount{$komb}\n"); }
    elsif($komb =~ /(\p{IsWord})x($n-2).'\-'.(\p{IsWord})/)
        {print(SKEEFRK "$komb\t$wordcount{$komb}\n"); }
    elsif($komb =~ /(\p{IsWord}).'\-'.(\p{IsWord}) x ($n-2)/)
        {print(SKEEFLK "$komb\t$wordcount{$komb}\n"); }
    elsif($komb =~ /^(\p{IsWord})x($n-1).'\-'/)
        {print(BEGINW "$komb\t$wordcount{$komb}\n"); }
    elsif($komb =~ /'\-'\(p{IsWord})x($n-1)$/)
        {print(EINDEW "$komb\t$wordcount{$komb}\n"); }
    else{print(NIEGEDEK "$komb\t$wordcount{$komb}\n"); }
    }
close(INVOER); close(WEERSKK); close(KWEERSK); close(NIEGEDEK);
close(VOORK); close(NAK); close(GEENK); close(SKEEFRK); close(SKEEFRK);
```

D.1.3 TeenstrGrep.pl

Plaas 3, 4, 5 en 6 letterkombinasies in KOMB#.dat (koppelteken op verskillende plekke). Bepaal die frekwensie van elke kombinasie. As die frekwnsie groter as een is, is daar moontlik 'n teenstrydigheid. Druk alle woorde met daardie kombinasie.

```
#!/usr/bin/perl
use utf8;

open(INVOER1,  "<:utf8", "KOMB3.dat")      or die ("Inv1!\n");       #komb met lg-verd
open(INVOER2,  "<:utf8", "KSONDER.txt")    or die ("Inv2!\n");       #lys sonder lg-verd
open(TEENSWrde,">:utf8", "TEENSWrde3.txt") or die ("Uitv1!\n");      #komb+frek+woorde
foreach $komb (<INVOER1>){
    $komb =~ tr/-//d;                                                #verwyder kts
    $komb =~ tr/\n$//d;                                              #verwyder "newline"
    $tel{$komb}++;
    }
foreach $komb(sort keys%tel) {
    if ($tel{$komb} > 2){                                            #vir komb > twee
        @woorde = grep /$komb/, <INVOER2>;                           #teenstrydigheid
        print(TEENSWrde "$komb\n@woorde\n");
        close(INVOER2);
        open(INVOER2,  "<:utf8", "KSONDER.txt")                      #invoerler weer oop
        substr($komb, 1, 0) = "-";                                   #(anders loop nie weer)
        @woorde = grep /$komb/, <INVOER2>;
        print(TEENSWrde "$komb\n@woorde\n");
        close(INVOER2);
        open(INVOER2,  "<:utf8", "KSONDER.txt")
        $komb =~ tr/-//d;
        substr($komb, -1, 0) = "-";
        @woorde = grep /$komb/, <INVOER2>;
        print(TEENSWrde "$komb\n@woorde\n");
        close(INVOER2);
        open(INVOER2,  "<:utf8", "KSONDER.txt")
        }
    }
close(INVOER1);
close(INVOER2);
close(TEENSTR);
close(WOORDE);
```

D.1.4 SortSonderKt.pl

Sorteer 'n woordelys sonder om koppeltekens in ag te neem.

```
#!/usr/bin/perl
use utf8;

open(INVOER,    "<:utf8",   "SGVerd.txt")     or die ("Invoer!!\n"); #invoerler
open(UITVOER,   ">:utf8",   "SGVerdSort.txt") or die ("Uitvoer1!!\n");
```

```
open(WrdKol,    ">:utf8",   "SGVerdKOL.txt")  or die ("Uitvoer!!\n");
while ($line = <INVOER>){
    chomp $line;
    $lineK = $line;                                                  #woord met koppeltekens
    $line =~ s/-//gi;                                                #woord sonder kts
    $lineS =$line;
    printf WrdKol ("%-39s %-41s\n", $lineS, $lineK);                 #wrde sonder en met kts in
    }                                                                #kolomme
close(WrdKol);
open(WrdKol,  "<:utf8", "SGVerdKOL.txt");
while ($str = <WrdKol>){
    push @Kolomme, $str;
    }
foreach $w (sort{substr($a,0,39) cmp substr($b,0,39)}@Kolomme){      #sorteer op wrde sonder kts
    push (@KOL, "$w");                                               #skep array van woorde
    }
for ($i=1; $i <= $#KOL; $i++){
    if (((substr($KOL[$i-1],0,39)) eq (substr($KOL[$i],0,39)))&
        (($KOL[$i-1] ne $KOL[$i]))){                                 #verskillende verdelings vir dieselfd
        print(UITVOER "$KOL[$i-1]$KOL[$i]\n");
        }
    elsif (($KOL[$i-1]) ne ($KOL[$i])){                              #verwyder duplikate
        $Wrd=$KOL[$i];
        $WrdKt = substr($Wrd, 40, 41);
        $WrdKt =~ tr/ //d;                                           #verwyder witspasie
        print (UITVOER "$WrdKt\n");
        }
    }
close(F);
close(UITVOER);
close(WrdKol);
```

D.2 VerdSGWrde.pl

Verdeel saamgestelde woorde in hul saamgestelde dele. Woorde word iteratief afgebreek totdat geen verdere verdeling moontlik is nie.

```
#!/usr/bin/perl
use utf8;
open(AL,        "<:utf8", "Verwysingslys.txt") or die("Invoer1!\n");  #VL in stygende lengte
open(LYS,       "<:utf8", "Toets.dat")         or die("Invoer2!\n");  #Woordelys om op te deel
open(AGTERV,    "<:utf8", "AgterVoegSort.txt") or die("Invoer3!\n");  #moontlike agtervoegsels
open(VOORV,     "<:utf8", "VoorVoegselsS.txt") or die("Invoer4!\n");  #moontlike voorvoegsels
open(BegW,      "<:utf8", "BeginWrdeW.txt")    or die("Invoer6!\n");  #kort wrde aan wrdbegin
open(EindW,     "<:utf8", "EindeWrdeSL.txt")   or die("Invoer7!\n");  #kort wrde aan wrdeinde
open(SGWrde,    ">:utf8", "SGWrde.txt")        or die("Uitv1!\n");    #Uitvoer: sg-woorde
open(GEENOpl,   ">:utf8", "GEENOpl.txt")       or die("Uitv2!\n");    #Uitvoer: oorspronklike wrde
open(WEERSK,    ">:utf8", "WEERSK.txt")        or die("Uitv3!\n");    #kt weerskante v konsonant
open(AlleWrde,  ">:utf8", "AlleWrde.txt")      or die("Uitv4!\n");    #alle woorde
while ($wrd = <AL>){
```

```
chomp ($wrd);
$lengte = length($wrd);
if ($lengte == 2){push @L2, $wrd;}      elsif ($lengte == 3){push @L3, $wrd;}
elsif ($lengte == 4){push @L4, $wrd;}   elsif ($lengte == 5){push @L5, $wrd;}
elsif ($lengte == 6){push @L6, $wrd;}   elsif ($lengte == 7){push @L7, $wrd;}
elsif ($lengte == 8){push @L8, $wrd;}   elsif ($lengte == 9){push @L9, $wrd;}
elsif ($lengte == 10){push @L10, $wrd;} elsif ($lengte == 11){push @L11, $wrd;}
elsif ($lengte == 12){push @L12, $wrd;} elsif ($lengte == 13){push @L13, $wrd;}
elsif ($lengte == 14){push @L14, $wrd;} elsif ($lengte == 15){push @L15, $wrd;}
elsif ($lengte == 16){push @L16, $wrd;} elsif ($lengte == 17){push @L17, $wrd;}
elsif ($lengte == 18){push @L18, $wrd;} elsif ($lengte == 19){push @L19, $wrd;}
elsif ($lengte == 20){push @L20, $wrd;} elsif ($lengte == 21){push @L21, $wrd;}
elsif ($lengte == 22){push @L22, $wrd;} elsif ($lengte == 23){push @L23, $wrd;}
elsif ($lengte == 24){push @L24, $wrd;} elsif ($lengte == 25){push @L25, $wrd;}
elsif ($lengte == 26){push @L26, $wrd;} elsif ($lengte == 27){push @L27, $wrd;}
elsif ($lengte == 28){push @L28, $wrd;} elsif ($lengte == 29){push @L29, $wrd;}
elsif ($lengte == 30){push @L30, $wrd;} elsif ($lengte == 31){push @L31, $wrd;}
elsif ($lengte == 32){push @L32, $wrd;} elsif ($lengte == 33){push @L33, $wrd;}
elsif ($lengte == 34){push @L34, $wrd;} }
while ($av = <AGTERV>){chomp ($av); push @AV, $av; }              #skep arrays
while ($vv = <VOORV>){chomp ($vv); push @VV, $vv; }
while ($beg = <BegW>){chomp ($beg); push @BEG, $beg; }
while ($eind = <EindW>){chomp ($eind); push @EIND, $eind; }
while ($tegno = <Tegno>){chomp ($tegno);push @TEGNO, $tegno; }
my $K = "bcdfghjklmnpqrstvwxz";
$N = 2;                                                           #kortste lettergreep
$n = 0;
while ($WRD = <LYS>){                                             #elke woord in lys
    chomp($WRD);
    $n++;  @WV = '';  $WV = '';  $WV1 = '';  $WV2 = '';  $WV3 = '';
    $WV2S = '';  $WW2 = '';  $WW3 = '';  @AVV = '';  @VVV = '';  $wrdd = '';
    @WRD = '';  @FWRD = '';  @FINLUS = "";  @FW = '';  $wfin = '';
    @WrdF = '';  $NWrd = '';  @NWRD = '';   $Nwoord = '';  $NNwoord = '';
    $t = 1;
    $f = 0;
    $NWRD[$t-1] = $WRD;
    $FWRD[$f] = $WRD;
    $L = length($WRD);
    foreach $woord (split (/\-|\=/, $WRD)){                       #verdeel by kt,
        chomp($woord);                                            #woorddele deur lus
        PROGRAM($woord);                                          #hoofprogram
        }
    $NWrd =  join ("-", @WRD);                                    #SG-dele saam met kts
    $NWrd =~ s/^.{1,1}//s;                                        #Verwyder kt vooraan
    $NWRD[$t] = $NWrd;
    if ($NWrd =~ m/($WRD)/){
        $woordF = $WRD;
        FIN($woordF);                                             #FINAAL-subroetine
        push @FWRD, $FINResultaat;
        $f++;
        if ($FWRD[$f] =~ m/($FWRD[$f-1])/){                       #geen verandering
            print GEENOpl "$FWRD[$f]\n";                          #oorspronklike woord
```

```
            print AlleWrde "$FWRD[$f]\n";
            }
        else {
            until ($FWRD[$f] =~ /($FWRD[$f-1])/){                #woorddele deur FIN
                foreach $woordF (split (/\-/, $FWRD[$f])){
                    chomp($woordF);
                    FIN($woordF);
                    push @FINLUS, $FINResultaat;
                    }
                $FFWrd = join ("-", @FINLUS);
                $FFWrd =~ s/^.{1,1}//s;
                push @FWRD, $FFWrd;
                $f++;
                @FINLUS = "";
                }
            print SGWrde "$FFWrd\n";
            print AlleWrde "$FFWrd\n";
            }
        }
    else{
        until ($NWRD[$t] =~ /($NWRD[$t-1])/){
            @WRD = '';
            foreach $woord (split (/\-/, $NWRD[$t])){            #verdeel by kt: elk deur lus
                chomp($woord);
                PROGRAM($woord);
                }
            $NWrd =  join ("-", @WRD);                           #voeg SG dele saam met kts
            $NWrd =~ s/^.{1,1}//s;                               #verwyder kt vooraan
            push @NWRD, $NWrd;
            $t++;
            }
        if ($NWrd =~ /\-[$K]\-/){
            print WEERSK "$NWrd\n";
            print AlleWrde "$NWrd\n";
            }
        else {
            $woordF = $NWrd;
            push @FWRD, $NWrd;
            $f++;
            until ($FWRD[$f] =~ /($FWRD[$f-1])/){
                foreach $woordF (split (/\-/, $NWrd)){
                    chomp($woordF);
                    FIN($woordF);
                    push @FINLUS, $FINResultaat;
                    }
                $FFWrd = join ("-", @FINLUS);
                $FFWrd =~ s/^.{1,1}//s;
                push @FWRD, $FFWrd;
                $f++;
                @FINLUS = "";
                }
           print SGWrde "$FFWrd\n";
```

```
                print AlleWrde "$FFWrd\n";
                }
            }
      }
#############################################################################################
sub PROGRAM{
    $Lw = length($woord);
    #@WRD = '';
    $woordV = $woord;
    VERDEEL($woordV);                                          #verdeel
    if (($WV =~ m/($woord)/)&($Lw > 5)){                       #oorspronklike wrd
        $woordA = $woord;
        AGTERVOOR($woordA);                                    #vv/av uit, verdeel res
        if (($WV3 =~ m/($woord)/)&($Lw > 9)){
            $woordK = $woord;
            KOMPLEKS($woordK);                                 #sonder langste beg/eind
            push @WRD, $WV2;                                   #woord vanaf KOMPLEKS
            }
        else{
            push @WRD, $WV1;                                   #woord vanaf AGTERVOOR
            }
        }
    else{                                                      #anders verdeel weer
        push @WRD, $WV;                                        #woord vanaf VERDEEL
        }
    }
#############################################################################################
sub VERDEEL{
    @WV = '';
    @BEGIN = '';
    @EINDE = '';
    @SGW = '';
    @SGWSort = '';
    @SGWS = '';
    $LwV = length($woordV);
    for ($j = $N; $j <= $LwV-$N; $j++){
        $LYS = 'L'.$j;
        $beg = substr($woordV, 0, $j);                         #lengte $j van begin
        $end = substr($woordV, -$j);                           #lengte $j van agter
        for($i = 0; $i <= $#$LYS; $i++) {                      #wrde van lengte $j
            if ($$LYS[$i] =~ m/($beg)/){                       #dieselfde as $beg
                push (@BEGIN, "$beg");
                }
            elsif ($$LYS[$i] =~ m/($end)/){                    #dieselfde as $end
                push (@EINDE, "$end");
                }
            }
        }
    for ($k = 1; $k <= $#BEGIN; $k++){
        for ($l = 1; $l <= $#EINDE; $l++){
            $SAAM = $BEGIN[$k].$EINDE[$l];
            if ((length($SAAM) >= $LwV-3) & (length($SAAM) <= $LwV)){
```

```
            if (length($SAAM) == $LwV){
                $SGWrd = $BEGIN[$k].'-'.$EINDE[$l];
                push @SGW, "$SGWrd";
                }
            }
        elsif ((length($SAAM) == $LwV-1) &                          #een verbindingsletter
            (substr($woordV, length($BEGIN[$k]), 1) =~ /s/)){
            $Verb = substr($woordV, length($BEGIN[$k]), 1);
            $SGWrd1 = $BEGIN[$k].$Verb.'-'.$EINDE[$l];
            $SGWrdS = $BEGIN[$k].$Verb.$EINDE[$l];
            if ($SGWrdS =~ m/($woordV)/){
                push @SGW, "$SGWrd1";
                }
            }
        elsif ((length($SAAM) == $LwV-2) &                          #twee verbindingsletters
            (substr($woordV, length($BEGIN[$k]), 2) =~ /en|ns|er/)){
            $Verb = substr($woordV, length($BEGIN[$k]), 2);
            $SGWrd2 = $BEGIN[$k].$Verb.'-'.$EINDE[$l];
            $SGWrdS = $BEGIN[$k].$Verb.$EINDE[$l];
            if ($SGWrdS =~ m/($woordV)/){
                push @SGW, "$SGWrd2";
                }
            }
        elsif ((length($SAAM) == $LwV-3) &                          #drie verbindingsletters
            (substr($woordV, length($BEGIN[$k]), 3) =~ /ens|ing/)){
            $Verb = substr($woordV, length($BEGIN[$k]), 3);
            $SGWrd3 = $BEGIN[$k].$Verb.'-'.$EINDE[$l];
            $SGWrdS = $BEGIN[$k].$Verb.$EINDE[$l];
            if ($SGWrdS =~ m/($woordV)/){
               push @SGW, "$SGWrd3";
               }
            }
        }
    }
if($#SGW == 0){
    $WV = $woordV;
    }
elsif ($#SGW == 1){                                                 #een weergawe - gebruik
    $WV = $SGW[1];
    }
else{                                                               #meer as 1 weergawe
    foreach $w (sort({$a cmp $b}@SGW)){                             #sorteer
        push (@SGWSort, "$w");
        }
    }
for ($i=1; $i <= $#SGWSort; $i++){
    if ($SGWSort[$i-1] ne $SGWSort[$i]){                            #verwyder duplikate
        $weerg=$SGWSort[$i];
        push @SGWS, "$weerg";
        }
    }
if ($#SGWS == 1){                                                   #een weergawe - gebruik
```

```
        $WV = $SGWS[1];
        }
    else{
        KOMB(@SGWS);
        $WV = $NW;
        }
    }
#############################################################################################
sub AGTERVOOR{
    @AVV = '';
    @VVV = '';
    @VERD = '';
    @VERDSort = '';
    @VERDS = '';
    $WV1 = '';
    foreach $av (@AV){                                          #ooreenstemmende avs
        $l = length($av);
        $wrdd = substr($woordA, -$l);
        if (($av =~ m/($wrdd)/)&($l < length($woordA))){
            push @AVV, $av;                                     #moontlike avs
            }
        }
    foreach $vv (@VV){                                          #ooreenstemmende avs
        $l = length($vv);
        $wrdd = substr($woordA, 0, $l);
        if (($vv =~ m/($wrdd)/)&($l < length($woordA))){
            push @VVV, $vv;                                     #lys van avs
            }
        }
    if ($#AVV > 0){
        for ($a=1; $a <= $#AVV; $a++){                          #as av(s) voorkom
            $l = length($AVV[$a]);                              #av van lank na kort
            $LLw = $Lw-$l;
            $woordV = substr($woordA, 0, $LLw);                 #woord sonder av
            VERDEEL($woordV);                                   #verdeel sonder av
            $wv1 = $WV.$AVV[$a];                                #haak av aan
            push @VERD, $wv1;
            $ll = length($wv1);                                 #lengte sonder '-'
            }
        }
    if ($#VVV > 0){                                             #as vv(s) voorkom
        $l = length($VVV[1]);                                   #vv van lank na kort
        $LLw = $Lw-$l;
        $woordV = substr($woordA, -$LLw);
        VERDEEL($woordV);
        if ($WV =~/($woordV)/){
            $woordK = $woordV;
            KOMPLEKS($woordK);
            $wv1 = $VVV[1].$WV2;
            push @VERD, $wv1;
            }
        else{
```

```
            $wv1 = $VVV[1].$WV;
            push @VERD, $wv1;
            }
        }
    if ($#VERD == 1){                                       #een weergawe - gebruik
        $WV1 = $VERD[1];
        }
    else{
        foreach $v (sort({$a cmp $b}@VERD)){                #sorteer
            push (@VERDSort, "$v");
            }
        for ($i=1; $i <= $#VERDSort; $i++){
            if ($VERDSort[$i-1] ne $VERDSort[$i]){          #verwyder duplikate
                $verd=$VERDSort[$i];
                push @VERDS, "$verd";
                }
            }
        if ($#VERDS == 1){                                  #een weergawe - gebruik
            $WV1 = $VERDS[1];
            }
        else{
            for($k=0, $k<=$#VERDS, $k++){
                $lll = length($VERDS[$k]);
                if ($lll > $Lw){
                    $WV1 = $VERDS[$k];
                    }
                }
            }
        }
    if (($#AVV == 0)&($#VVV == 0)){
        $WV1 = $woord;
        }
    }
##################################################################################################
sub KOMPLEKS{
    @BEGIN = '';
    @EINDE = '';
    $WV2 = '';
    $Lk = length($woordK);
    for ($j = $N; $j <= $Lk-$N; $j++){
        $LYS = 'L'.$j;
        $beg = substr($woordK, 0, $j);                      #string lengte $j van voor
        $end = substr($woordK, -$j);                        #string lengte $j van agter
        for($i = 0; $i <= $#$LYS; $i++) {                   #lys met wrde van lengte $j
            if ($$LYS[$i] =~ m/($beg)/){                    #woord van lengte $j = $beg
               push (@BEGIN, "$beg");
               }
            if ($$LYS[$i] =~ m/($end)/){
                push (@EINDE, "$end");
                }
            }
        }
```

```
    $LangsteBeg = $BEGIN[$#BEGIN];
    $LangsteEind = $EINDE[$#EINDE];
    $lb = length($LangsteBeg);                                  #lengte langste beginwrd
    $le = length($LangsteEind);                                 #lengte langste eindewrd
    $Lb = $Lk-$lb;                                              #sonder langste beginwrd
    $Le = $Lk-$le;                                              #sonder langste eindewrd
    $woordV = substr($woordK, -$Lb);                            #sonder langste beginwrd
    $woordA = substr($woordK, 0, $Le);                          #sonder langste eind
    VERDEEL($woordV);
    $LWV = length($WV);
    if (($LWV > $Lb)&($lb > 3)){                                #as suksesvol verdeel
        $WV2 = $LangsteBeg.'-'.$WV;
        }
    elsif ($le > 0){
        $woordV = $woordA;
        VERDEEL($woordV);
        if ($WV =~ $woordA){
            $WV2 = $WV.$LangsteEind;
            }
        else{
            $WV2 = $WV.'-'.$LangsteEind;
            }
        }
    else{
        $WV2 = $woordK;
        }
    }
#############################################################################################
sub FIN{
@SGWS = '';
@RWB = '';
@RWE = '';
@BW = '';
@EW = '';
@FW = '';
$Res = '';
$Lw = length($woordF);
foreach $beg (@BEG){                                            #elke beginwrd
    $lb = length($beg);
    $Begw = substr($woordF, 0, $lb);
    $lbw = length($Begw);
    if (($beg =~ m/($Begw)/)&($lb==$lbw)&($lb<$Lw)&($lb > 0)){
        $lbf = length($Begw);
        $BW = $Begw;
        push @BW, $Begw;
        }
    }
foreach $eind (@EIND){
    $le = length($eind);
    $Eindw = substr($woordF, -$le);
    $lew = length($Eindw);
    if (($eind =~ m/($Eindw)/)&($le==$lew)&($le<$Lw)){
```

```
        $lef = length($Eindw);
        $EW = $Eindw;
        push @EW, $Eindw;
        }
    }
if (($#BW > 0)&($#EW == 0)){
    for ($b = 1; $b <= $#BW; $b++){
        @RWB = "";
        $lb = length($BW[$b]);
        $ResB = substr($woordF, $lb, $Lw);
        $lrb = length($ResB);
        $LYS = 'L'.$lrb;
        for ($i = 0; $i<= $#$LYS; $i++) {
            if ($$LYS[$i] =~ m/($ResB)/){
                $rwb = $$LYS[$i];
                    push @RWB, $rwb
                    }
                }
        if ($#RWB > 0){
            $wfinB = $BW[$b].'-'.$ResB;
            push @FW, $wfinB;
            }
        elsif ($#RWB == 0){
            @AVV = "";
            foreach $av (@AV){                                  #elke av in av-lys
                $la = length($av);
                $wrdd = substr($ResB, -$la);                    #woord sonder av
                if (($av =~ m/($wrdd)/)&($la < length($ResB))){
                    push @AVV, $av;                             #lys van moontlike avs
                    }
                }
            if ($#AVV > 0){
                for ($a=1; $a <= $#AVV; $a++){                  #as av(s) voorkom
                    @EW = "";
                    $lav = length($AVV[$a]);                    #avs van lank na kort
                    $lsa = $lrb-$lav;                           #lengte sonder av
                    $woordSA = substr($ResB, 0, $lsa);          #wrd sonder av
                    foreach $eind (@EIND){                      #verdeel sonder av
                        $le = length($eind);
                        if (($eind =~ m/($woordSA)/)&($le == $lsa)){
                            push @EW, $eind;
                           }
                        }
                    if (($#EW > 0)&($lav + $lsa == $lrb)){
                        $wfinB = $BW[$b]."-".$EW[1].$AVV[$a];   #haak av aan
                        push @FW, $wfinB;
                        }
                    else{
                        push @FW, $woordF;
                        }
                    }
                }
```

```
            else{
                $wfinB = $woordF;
                push @FW, $wfinB;
                }
            }
        }
    }
elsif (($#EW > 0)&($#BW == 0)){
    @RWE = "";
    for (my $e = 1, $e <= $#EW, $e++){
        $le = length($EW[$e]);
        $ResE = substr($woordF, 0, $Lw-$le);
        $lre = length($ResE);
        $LYS = 'L'.$lre;
        for ($j = 0; $j <= $#$LYS; $j++){
            if ($$LYS[$j] =~ m/($ResE)/){
                $rwe = $$LYS[$j];
                push @RWE, $rwe;
                }
            }
        }
    if ($#RWE > 0){
        $wfinE = $ResE.'-'.$EW;
        push @FW, $wfinE;
        }
    elsif ($#RWE == 0){
        $wfinE = $woordF;
        push @FW, $wfinE;
        }
    }
elsif (($#BW > 0)&($#EW > 0)){
        for (my $b = 1; $b <= $#BW; $b++){
            @RWB = "";
            for (my $e = 1; $e <= $#EW; $e++){
                $lb = length($BW[$b]);
                $le = length($EW[$e]);
                if ($lb + $le == $Lw){
                    $wfinBE = $BW[$b].'-'.$EW[$e];
                    push @FW, $wfinBE;
                    }
                else {
                    $ResB = substr($woordF, $lb, $Lw);
                    $lrb = length($ResB);
                    $ResE = substr($woordF, 0, $Lw-$le);
                    $lre = length($ResE);
                    $LYS = 'L'.$lrb;
                    for ($i = 0; $i<= $#$LYS; $i++) {
                        if ($$LYS[$i] =~ m/($ResB)/){
                            $rwb = $$LYS[$i];
                            push @RWB, $rwb
                            }
                        }
```

```
                    $LYS = 'L'.$lre;
                    for ($j = 0; $j<= $#$LYS; $j++) {
                        if ($$LYS[$j] =~ /($ResE)/){
                            $rwe = $$LYS[$j];
                            push @RWE, $rwe;
                            }
                        }
                    if (($#RWB > 0)&($lb + $lrb == $Lw)){
                        $wfinBE = $BW[$b].'-'.$rwb;
                        push @FW, $wfinBE;
                        }
                    elsif (($#RWE > 0)&($le + $lre == $Lw)){
                        $wfinBE = $rwe.'-'.$EW[$e];
                        push @FW, $wfinBE;
                        }
                    elsif (($#RWE == 0)&($#RWB == 0)){
                       $wfinBE = $woordF;
                       push @FW, $wfinBE;
                       }
                    }
                }
            }
        }
elsif (($#BW == 0)&($#EW == 0)){
    push @FW, $woordF;
    }
    if ($#FW > 1){
        @SGWS = @FW;
        KOMB(@SGWS);
        $FINResultaat = $NW;
        }
    else{
        $FINResultaat = $FW[1];
        }
    }
##################################################################################################
sub KOMB{
    for ($m = 1; $m < $#SGWS; $m++){                                    #meer as een wg - komb
        $NW = "";
        $s = 0;
        @wrdk1 = split //, $SGWS[$m];                                   #karakters van wg
        @wrdk2 = split //, $SGWS[$m+1];                                 #kar van volgende wg
        for (my $i=0; $i <= $#wrdk1; $i++){                             #kar van eerste wg
            if ($wrdk1[$i] =~ /($wrdk2[$i+$s])/){                       #kar in beide, in wrd
                $NW = $NW.$wrdk2[$i+$s];
                }
            elsif (($wrdk1[$i] =~ /\-/) &
                ($wrdk2[$i+$s] =~ /\w/)){
                $NW = $NW.$wrdk1[$i];
                $s--;
                }
            elsif (($wrdk1[$i] =~ /\w/) &
```

```
                ($wrdk2[$i+$s] =~ /\-/)){
                $NW = $NW.$wrdk2[$i+$s].$wrdk1[$i];
                $s++;
                }
            }
            $SGWS[$m+1] = $NW;
        }
    }
##################################################################################################
close(LYS);
close(AL);
close(SGWrde);
close(GEENOpl);
close(WEERSK);
```

D.3 Neurale netwerke

Aangesien die enkodering herhaaldelik gebruik word en baie spasie beslaan, is daar telkens 'n verwysing na Afdeling D.7.1 op bladsy 166 waar dit getoon word.

D.3.1 GenAfrigDataNN.pl

Genereer 'n matriks waarvan die rye uit gekodeerde vensters met 268 karakters bestaan en 'n teikenvektor met 268 rye wat elk uit 'n enkele nul of een bestaan.

```
#!/usr/bin/perl
#use strict;
use utf8;

open(DATA,         "<:utf8", "LG_Afrig.txt") or die "Invoer!";       #woorde met lgverdeling
open(UNIEKFRSRT,   ">:utf8", "ToetsUniekFrekSrt.txt");               #ven + tei gesorteer
open(NNAfrigVen,   ">:utf8", "RLLG1kVen.txt");                       #vensters
open(NNAfrigTei,   ">:utf8", "RLLG1kTei.txt");                       #teikens
$vl=5;  $vr=3;                                                       #venster links en regs
$vt=$vl+$vr;                                                         #totale venster
$AfrigTeiken="";
while($woord = <DATA>){                                              #elke woord in invoer
    $wl=length($woord);                                              #woordlengte (met kts)
    $teiken="";                                                      #teikenvektor
    $wrdsp="";                                                       #woord met spasies
    $sp=" ";                                                         #spasie
    $s=0;                                                            #skuifregister
    $w="";                                                           #woord sonder kt
    @wrdkar = split //, lc($woord);                                  #karakters in array
    for (my $i=0; $i < $wl-$s-1; $i++){                              #karakters in woord
        if(($wrdkar[$i+$s] =~ /\p{IsWord}|\'|\=/)&
            ($wrdkar[$i+$s+1] =~/\-/)){                              #kt na letter|'|=
            $t[$i+$s]=1;                                             #teiken = 1
            $w=$w.$wrdkar[$i+$s];                                    #letter in nuwe wrd
```

```
            $teiken=$teiken.$t[$i+$s];                          #teiken in teikenry
            $s++;                                               #slaan '-' oor
            }
        elsif(($wrdkar[$i+$s] =~ /\p{IsWord}/)&
            ($wrdkar[$i+$s+1] =~/\n/)){                         #laaste letter
            $w=$w.$wrdkar[$i+$s];                               #letter in woord
            }
        else{                                                   #letter na letter
            $t[$i+$s]=0;                                        #teiken = 0
            $w=$w.$wrdkar[$i+$s];                               #letter in woord
            $teiken=$teiken.$t[$i+$s];                          #teiken in teikenry
            }
    }
    @teiken = split //, $teiken;
    $AfrigTeiken=$AfrigTeiken.$teiken;                          #teikens
    $wrdsp=($sp x ($vl-1)).$w.($sp x ($vr-1));                  #spasies
    $lwrdsp = length($wrdsp);                                   #lengte met spasies
    for(my $j=0; $j <= $lwrdsp-$vt; $j++){
        $venster=substr($wrdsp, $j, $vt);                       #genereer vensters
        push (@vensters, "$venster\n");
        @kar=split //, $venster;                                #karakters in array
        $invoer="";                                             #gekodeerde invoer
        for (my $k=0; $k < $vt; $k++){                          #karakters in venster
            [ENKODERING soos op bladsy 166]
            print KODES "$kode\n";
            $invoer=$invoer.$kode;                              #invoerry vir venster
            }
        push (@pare, "$invoer@teiken[$j]\n");
        }
    }
foreach $paar (@pare){
    $wordcount{$paar}++;                                        #tel vensters
    }
foreach $paar (sort keys(%wordcount)){                          #sorteer
    push @UniekFr, "$paar $wordcount{$paar}\n";
    }
@sorted = sort {substr($b,-3,3) cmp substr($a,-3,3)} @UniekFr;
foreach $ventei (@sorted){
    $fin = substr($ventei,0,$vt*46*2+2);
    $frek = substr($ventei, -3);
    print UNIEKFRSRT "$fin\t$frek";                             #gesorteer vlg frek
    $afrigven = substr($fin,0,$vt*46*2);
    $afrigtei = substr($fin,-2);
    print NNAfrigVen "$afrigven\n";                             #vensters vir afrigting
    print NNAfrigTei "$afrigtei";                               #teikens vir afrigting
    }
close(DATA);
close(UNIEKFRSRT);
close(NNAfrigVen);
close(NNAfrigTei);
```

D.3.2 ToetsNN.pl

Gebruik gewigte van afgerigte neurale netwerk om 'n woordelys in lettergrepe te verdeel, teen die korrekte verdeling te toets en statistiek te bereken.

```
#!/usr/bin/perl
use utf8;

open(DATA,      "<:utf8", "LG_Toets.txt");                          #woorde met lgverdeling
open(NNUIT,     ">:utf8", "SG5k100_NNUIT.txt");                     #lgverdeling volgens NN

open(InvoerGewigte,    "<IW5k100.txt") or die "Inv Gewigte!";
open(VerborgeBeladings,"<HB5k100.txt") or die "Verb Beladings!";
open(VerborgeGewigte,  "<HW5k100.txt") or die "Verb Gewigte!";
open(UitvoerBeladings, "<OB5k100.txt") or die "Uitv Beladings!";

open(NNWRDES,   ">:utf8", "SG5k100_NNWrdes.txt");                   #waardes (misse en foute)
open(NNREGWEG,  ">:utf8", "SG5k100_NNREGWEG.txt");                  #woorde kt reg weggelaat
open(NNREGIN,   ">:utf8", "SG5k100_NNREGIN.txt");                   #woorde reg verdeel
open(NNMIS,     ">:utf8", "SG5k100_NNMIS.txt");                     #kt gemis
open(NNFOUT,    ">:utf8", "SG5k100_NNFOUT.txt");                    #kt op verkeerde plek
open(NNWKMK,    ">:utf8", "SG5k100_NNWKMK.txt");                    #kt weerskante van medekl
open(STAT,      ">:utf8", "SG5k100_NNSTAT.txt");                    #statistiek aandui
open(UIT,       ">:utf8", "Uit.txt");                               #NN se uitvoer

while ($VB = <VerborgeBeladings>){
    $VerbB = trim($VB);
    push(@VerbB, $VerbB);
    }
while ($UitvB = <UitvoerBeladings>){
    $UB =trim($UitvB);
    push @UitvB, $UB;
    }
while ($line = <VerborgeGewigte>){
    $VG = trim($line);
    @VerbG = split(/\s+/, $VG);                                     #breek op by spasies
    }
while (<InvoerGewigte>){
    push @InvG, [ split ];
    }
my $C = "bcdfghjklmnpqrstvwxz";
$vl=4;   $vr=4;                                                     #venster links en regs
$vt=$vl+$vr;                                                        #totale venster
$woord1= "";
$n=0;                                                               #woorde tel
$g=0;                                                               #geleenthede tel
$totg = 0;
$NNwreg=0;                                                          #wrde reg
$NNwverkeerd=0;                                                     #wrde verkeerd
$NNwregin = 0;                                                      #wrde met kt(s) reg in
$NNwregweg=0;                                                       #wrde met kt(s) reg weg
$NNwmis=0;                                                          #wrde met kts gemis
```

```
$NNwfout=0;                                                    #wrde met kt-foute
$NNgreg=0;                                                     #vgeleenthede reg
$NNgverkeerd;                                                  #vgeleenthede verkeerd
$NNktregin = 0;                                                #kt reg ingeplaas
$NNktregweg=0;                                                 #Kt reg weggelaat
$NNgverkeerd=0;                                                #geleenthede verkeerd
$NNgfout=0;                                                    #geleenthede fout
$NNgmis=0;                                                     #geleenthede gemis
$telNNuitv=0;
while($woord1 = <DATA>){                                       #elke woord in invoer
    $telNNuitv=0;                                              #teller vir @UITVL2
    $n++;
    @UITVL2="";
    $wl=length($woord1);                                       #woordlengte (met kts)
    $teiken="";                                                #teikenvektor
    $wrdsp="";                                                 #woord met spasies
    $sp=" ";                                                   #spasie
    $s=0;                                                      #skuifregister
    $w="";                                                     #woord sonder kt
    @wrdkar = split //, lc($woord1);                           #karakters in array
##############################################################################Skep teiken
    for (my $i=0; $i < $wl; $i++){                             #karakters in woord
        if(($wrdkar[$i+$s] =~ /\w|\'|\=/)&
            ($wrdkar[$i+$s+1] =~/\-/)){                        #kt na letter/'/=
            $t[$i+$s]=1;                                       #teiken = 1
            $w=$w.$wrdkar[$i+$s];                              #letter in wrd (sonder -)
            $s++;                                              #skuif: slaan '-' oor
            }
        elsif(($wrdkar[$i+$s] =~ /\w/)&
            ($wrdkar[$i+$s+1] =~/\n/)){                        #laaste letter
            $w=$w.$wrdkar[$i+$s];
            }
        else{                                                  #letter na letter
            $t[$i+$s]=0;                                       #teiken = 0
            $w=$w.$wrdkar[$i+$s];                              #letter in woord
            }
        }
        @wrdsonder = split //, $w;
        $g = $#wrdsonder-1;
        $totg = $totg + $g;
        $wrdsp=($sp x ($vl-1)).$w.($sp x ($vr-1));             #spasies voor en na wrd
        $lwrdsp = length($wrdsp);                              #woord met spasies
        $NNwrdmet="";                                          #afbreking met NN
        $laastelet = $wrdsonder[$#wrdsonder];
#####################################################################Genereer vensters & enkodeer
        for(my $j=0; $j < $lwrdsp-$vt+1; $j++){                #begin VENSTERS
            $venster=substr($wrdsp, $j, $vt);                  #genereer vensters
            @kar=split //, $venster;                           #vensterkarakters in array
                $venkode="";                                   #gekodeerde invoer
            $kode="";
            for (my $k=0; $k < $vt; $k++){                     #elke karakter in venster
                [ENKODERING SOOS OP BLADSY 166]
```

```
                }
            $venkode=$venkode.$kode;                                    #invoerry
            }
##############################################################################Bepaal NN uitvoer
    @Invoer = split(/\s+/, $venkode);
    for $c (0 .. $#InvG){                                               #verb neurone
        $Som1[$c] = 0;
        for $d (0 .. $#{$InvG[$c]}){                                    #invoergewigte
            $InvL1[$d]=$Invoer[$d]*$InvG[$c][$d];                       #invoer x gewig
            $Som1[$c] = $Som1[$c]+$InvL1[$d];                           #sommeer
            }
        $Net1[$c]=$Som1[$c] + $VerbB[$c];
        $UitvL1[$c] = 1/(1+exp(-($Net1[$c])));                          #logsig oordragf
        }                                                               #Einde laag 1
        $Som2 = 0;
        for $l (0 .. $#VerbG){                                          #verst gewigte
            $InvL2[$l] = $UitvL1[$l] * $VerbG[$l];                      #UitvL1 x VerbG.
            $Som2 = $Som2 + $InvL2[$l];
            }
        $Net2 = $Som2 + $UitvB[0];
        $UitvL2 = 1/(1+exp(-($Net2)));                                  #logsig oordragf
        $r = sprintf("%.2f", $UitvL2);
        push (@UITVL2, $r);
        if ($UitvL2 >= 0.5 ){
        $NNwrdmet = $NNwrdmet.$wrdsonder[$j]."-";
        #$kt++;
        v}
        elsif($UitvL2 < 0.5){
            $NNwrdmet = $NNwrdmet.$wrdsonder[$j];
        }
    $telNNuitv++;
    }                                                                   #Einde vensters
    $NNwrdmet = $NNwrdmet.$laastelet;                                   #Laaste letter
    print NNUIT "$NNwrdmet";
    $wl1=length($woord1);
    $wl2=length($NNwrdmet);
    @w1k = split //, $woord1;                                           #woord1 karakters
    @w2k = split //, $NNwrdmet;                                         #woord2 karakters

    $NNs=0;                                                             #skuifregister
    $NNtots=0;                                                          #totale skuif
    $NNSIG="";                                                          #signatuur
    $NNtelSF=0;                                                         #foute
    $NNtelGF=0;                                                         #misse
    $NNNW=""                                                            #woord opgebou
    $NNW="";                                                            #NN-waardes
    $telNNuit = 0;
##############################################################################Tel geleenthede
    for (my $i=0; $i < $wl1; $i++){
        if((($w1k[$i]=~/\w/) & ($w2k[$i+$NNs]=~/\w/))|
            (($w1k[$i]=~/\'/) & ($w2k[$i+$NNs]=~/\'/))|
            (($w1k[$i]=~/\=/) & ($w2k[$i+$NNs]=~/\=/))){                #karakters ooreen
```

```
            $telNNuit++;                                                #NN uitvoere
            $NNNW=$NNNW."$w1k[$i]";
            $NNW=$NNW."$w1k[$i]";
            $NNgreg++;
            }
        elsif(($w1k[$i] eq "-") & ($w2k[$i+$NNs] eq "-")){              #kt reg in
            $NNNW=$NNNW."$w1k[$i]";
            $NNW=$NNW."$w1k[$i]"."($UITVL2[$telNNuit])";
            $NNktregin++;
            $NNgreg++;
            }
        elsif($w1k[$i] eq "-" & $w2k[$i+$NNs] =~ /\w/){                 #kt weggelaat
            $NNNW=$NNNW."!";                                            #! vir mis
            $NNW=$NNW."!"."($UITVL2[$telNNuit])";
            $NNgreg--;
            $NNs--;                                                     #vorige wrd2 kar
            $NNtots++;                                                  #totale skuif
            $NNgmis++;                                                  #mis
            $NNgverkeerd++;                                             #fout
            }
        elsif($w1k[$i] =~ /\w/ & $w2k[$i+$NNs] eq "-"){                 #fout
            $NNNW=$NNNW."*$w1k[$i]";
            $NNW=$NNW."*"."($UITVL2[$telNNuit])"."$w1k[$i]";
            $NNs++;                                                     #vlg kar in wrd2
            $NNgreg--;
            $NNtots++;
            $NNgfout++;                                                 #fout
            $NNgverkeerd++;
            $telNNuit++;
            }
        }
##############################################################################Tel woorde
    if(($NNtots == 0)&($woord1 =~ /\-+/)){
        $NNwreg++;                                                      #Woord reg
        $NNwregin++;                                                    #sg-woord reg
        print NNREGIN "$NNNW\n";
        print NNWRDES "$NNW\n";
        }
    elsif(($NNtots == 0)&($woord1 !~ /\-/)){
    $NNwreg++;                                                          #woord reg
    $NNwregweg++;                                                       #reg onverdeel
    print NNREGWEG "$NNNW\n";
    print NNWRDES "$NNW\n";
    }
    elsif($NNtots == -$NNs){
        $NNwmis++;                                                      #woord met mis
        $NNwverkeerd++;                                                 #woord met fout
        print NNMIS "$NNNW\n";
        print NNWRDES "$NNW\n";
        }
    else{
        $NNwfout++;                                                     #woord met fout
```

```
        $NNwverkeerd++;                                                 #woord verkeerd
        print NNFOUT "$NNNW\n";
        print NNWRDES "$NNW\n";
        if($NNNW =~ /\*[$C]\-|\-[$C]\*/){
            print NNWKMK "$NNNW\t\t$NNW\n";
            }
        }
    }                                                                   #einde woorde
$NNgreg=$totg - $NNgverkeerd;
$NNktregweg = $NNgreg - $NNktregin;
#############################################################################Persentasies
$NNpwreg = sprintf("%.2f",$NNwreg/$n*100);                              #% woorde reg
$NNpwverkeerd = sprintf("%.2f",$NNwverkeerd/$n*100);                    #% woorde verkeerd
$NNpwregin = sprintf("%.2f",$NNwregin/$n*100);                          #% wrde reg verdeel
$NNpwregweg = sprintf("%.2f",$NNwregweg/$n*100);                        #% woorde reg
$NNpwmis  = sprintf("%.2f",$NNwmis/$n*100);                             #% woorde met misse
$NNpwfout  = sprintf("%.2f",$NNwfout/$n*100);                           #% woorde met foute

$NNpgreg = sprintf("%.2f",$NNgreg/$totg*100);                           #% kts reg ingeplaas
$NNpgverkeerd = sprintf("%.2f",$NNgverkeerd/$totg*100);                 #% gel verkeerd
$NNpktregin = sprintf("%.2f",$NNktregin/$totg*100);                     #% kts reg ingeplaas
$NNpktregweg = sprintf("%.2f",$NNktregweg/$totg*100);                   #% kts reg weggelaat
$NNpgmis  = sprintf("%.2f",$NNgmis/$totg*100);                          #% geleenthede gemis
$NNpgfout  = sprintf("%.2f",$NNgfout/$totg*100);                        #% geleentheidsfoute

print STAT  "\n\n
\t\t\t\t\tNEURALE NETWERK \n\n
Totale aantal woorde:\t\t\t$n\n
Aantal woorde reg:\t\t\t $NNwreg \t $NNpwreg%
Aantal woorde verkeerd:\t\t\t $NNwverkeerd \t $NNpwverkeerd%\n
Aantal saamgestelde woorde reg:\t\t $NNwregin \t $NNpwregin%
Aantal enkelvoudige woorde reg:\t\t $NNwregweg \t $NNpwregweg%
Aantal woorde met lg gemis:\t\t $NNwmis \t $NNpwmis%
Aantal woorde met foute:\t\t $NNwfout \t $NNpwfout% \n\n

Totale aantal verdelingsgeleenthede:\t $totg\n
Aantal geleenthede reg:\t\t\t $NNgreg \t $NNpgreg%
Aantal geleenthede verkeerd:\t\t $NNgverkeerd \t $NNpgverkeerd%\n
Aantal koppeltekens reg ingeplaas:\t $NNktregin\t$NNpktregin%
Aantal koppeltekens reg weggelaat:\t $NNktregweg\t$NNpktregweg%
Aantal verdelingsgeleenthede gemis:\t $NNgmis \t $NNpgmis%
Aantal verdelingsgeleentheidsfoute:\t $NNgfout \t $NNpgfout% ";

close(InvoerGewigte);
close(VerborgeBeladings);
close(VerborgeGewigte);
close(UitvoerBeladings);
close(NNUIT);
close(NNWRDES);
close(NNREG);
close(NNMIS);
close(NNFOUT);
```

```
close(NNWKMK);
close(NNFS);
close(STAT);
close(DATA);
```

D.4 Beslissingsbome

D.4.1 Genereer afrigtingsdata

Genereer vensters (posisies 1-8) en teikenwaarde (posisie 9) in een ry. Gebruik 'n '+' as plekhouer in plaas van 'n spasie.

```
#!/usr/bin/perl
use utf8;
open(DATA,       "<:utf8", "LG_Afrig.txt") or die "Invoer!";        #woorde met lgverdeling
open(UNIEKFRSRT, ">:utf8", "RNDLYS1UniekFrekSrt.txt");              #unieke ven/tei frek sort
open(BBAfrig,    ">:utf8", "BBAfrig1Nulle.txt");                    #venster/teiken in een ry
$vl=4;  $vr=4;                                                      #venster links en regs
$vt=$vl+$vr;                                                        #totale venster
$AfrigTeiken="";
print BBAfrig "L1 L2 L3 L4 L5 L6 L7 L8 TT\n";                       #verandelikes
while($woord = <DATA>){                                             #elke woord in invoer
    #chomp($woord);
    $wl=length($woord);                                             #woordlengte (met kts)
    $teiken="";                                                     #def vektor vir teiken
    $wrdsp="";                                                      #woord met spasies
    $sp="+";                                                        #spasie
    $s=0;                                                           #skuif
    $w="";                                                          #nuwe woord sonder -
    @wrdkar = split //, lc($woord);                                 #karakters in array
    for (my $i=0; $i < $wl-$s-1; $i++){                             #karakters in woord
        if(($wrdkar[$i+$s] =~ /\p{IsWord}|\'|\=/)&
            ($wrdkar[$i+$s+1] =~/\-/)){                             #'-' na letter/'/=
            $t[$i+$s]=1;                                            #teiken = 1
            $w=$w.$wrdkar[$i+$s];                                   #letter in nuwe wrd
            $teiken=$teiken.$t[$i+$s];                              #teiken in teikenry
            $s++;                                                   #slaan '-' oor
            }
        elsif(($wrdkar[$i+$s] =~ /\p{IsWord}/)&
            ($wrdkar[$i+$s+1] =~/\n/)){                             #laaste letter
           # print "$wrdkar[$i+$s]\n";
            $w=$w.$wrdkar[$i+$s];                                   #letter in nuwe woord
            }
        else{                                                       #letter volg letter
            $t[$i+$s]=0;                                            #teiken = 0
            $w=$w.$wrdkar[$i+$s];                                   #letter in nuwe woord
            $teiken=$teiken.$t[$i+$s];                              #teiken in teikenry
            }
    }
    print "$w\n";
```

```
    @teiken = split //, $teiken;
    $AfrigTeiken=$AfrigTeiken.$teiken;
    $wrdsp=($sp x ($vl-1)).$w.($sp x ($vr-1));                    #spasies
    $lwrdsp = length($wrdsp);                                     #lengte met spasies
    for(my $j=0; $j <= $lwrdsp-$vt; $j++){
        $venster=substr($wrdsp, $j, $vt);                         #genereer vensters
        @kar=split //,$venster;                                   #vensterkarakters
        }
    $invoer="";                                                   #gekodeerde invoer
    $kode="";
    for (my $k=0; $k < $vt; $k++){                                #elke vensterkaraker
        [ENKODERING soos op bladsy 170]
        $invoer=$invoer.$kode;                                    #invoerry vir venster
        }
    }
foreach $ventei (@ventei){
    $wordcount{$ventei}++;                                        #frek van ventei pare
    }
foreach $ventei (sort keys(%wordcount)){
    push @UniekFr, "$ventei$wordcount{$ventei}\n";
    }
@sorted = sort {substr($b,-3,3) cmp substr($a,-3,3)} @UniekFr;
foreach $ventei (@sorted){
    $fin = substr($ventei,0,17);
    $fr = substr($ventei, -2);
    print UNIEKFRSRT "$fin\t$fr";
    print BBAfrig "$fin\n";
    }
#################################################################################################
close(DATA);
close(UNIEKFR);
close(UNIEKFRSRT);
close(BBAfrig);
```

D.4.2 BBReelsRegmaak.pl

Die reëls van 'n afgerigte beslissingsboom word aangepas sodat dit in 'n Perl-program gebruik kan word.

```
#!/usr/bin/perl
use utf8;
open(DATA, "<:utf8", "LG_AfrigEntRules.txt") or die "INVOER!";       #Invoer - BB-reels
open(RREG, ">:utf8", "LG_AfrigEntRulesReg.pl");                      #BB-reels vir Perl
while($lyn = <DATA>){
    $k = substr($lyn,8,9);
    if($lyn =~ /class/){
        substr($lyn, 4, 0) = '$';
        print RREG "$lyn";
        }
    elsif($lyn =~/if/){
        print RREG "$lyn";
```

```
        }
    elsif($lyn =~ /\$/){
        $lyn =~ s/\$//g;
        substr($lyn, 7, 0) = '$';
        $lyn =~ s/\|\|/\|/g;
        if(($k == 10)|($k == 11)|($k == 12)){
            $v = substr($lyn,15,2);
            }
        else{
            $v = substr($lyn,14,2);
            }
        $lyn =~ s/$v/"$v"/g;
        $lyn =~ s/\=\=/eq/g;
        print RREG "$lyn";
        }
    elsif($lyn =~ s/\&\&/\&/g){
        print RREG "$lyn";
        }
    elsif($lyn =~ /\(|\)|\{|\}/){
        print RREG "$lyn";
        }
    elsif($lyn =~ /Terminal/){
        print RREG "#$lyn";
        }
    }
print RREG "1;";
close(DATA);
close(RREG);
```

D.4.3 ToetsBB.pl

Gebruik reëls van die afgerigte BB om 'n woordelys in lettergrepe te verdeel, teen die korrekte verdelings te toets en statistiek te bereken. Die enkodering word in Afdeling D.7.2 op bladsy 168 getoon.

```
#!/usr/bin/perl
use utf8;

open(DATA,        "<:utf8", "LG_Toets.txt");                    #Invoer
open(BBUIT,       ">:utf8", "LGAfr55_BBUIT.txt");               #woorde volgens BB
open(BBREGWEG,    ">:utf8", "LGAfr55_BBREGWEG.txt");            #ev-woorde reg
open(BBREGIN,     ">:utf8", "LGAfr55_BBREGIN.txt");             #sg-woorde reg
open(BBMIS,       ">:utf8", "LGAfr55_BBMIS.txt");               #mis
open(BBFOUT,      ">:utf8", "LGAfr55_BBFOUT.txt");              #fout
open(BBWKMK,      ">:utf8", "LGAfr55_BBWKMK.txt");              #kt weerskante konsonant
open(STAT,        ">:utf8", "LGAfr55_BBSTAT.txt");              #statistiek

my $C = "bcdfghjklmnpqrstvwxz"
$vl=5;  $vr=5;                                                  #venster links en regs
$vt=$vl+$vr;                                                    #totale venster
$woord1= "";
```

```
$n=0;                                                    #woorde tel
$g=0;                                                    #geleenthede tel
$totg = 0;
$BBwreg=0;                                               #woorde reg
$BBwverkeerd=0;                                          #woorde verkeerd
$BBwregin = 0;                                           #wrde met kt(s) reg in
$BBwregweg=0;                                            #wrde met kt(s) reg weg
$BBwmis=0;                                               #woorde met kts gemis
$BBwfout=0;                                              #woorde met kts fout

$BBgreg=0;                                               #geleenthede reg
$BBgverkeerd;                                            #geleenthede verkeerd
$BBktregin = 0;                                          #kt reg in
$BBktregweg=0;                                           #kt reg weg
$BBgfout=0;                                              #geleenthede fout
$BBgmis=0;                                               #geleenthede mis
$telBBuitv=0;

while($woord1 = <DATA>){                                 #elke woord in invoer
    $telBBuitv=0;
    $n++;
    $wl=length($woord1);                                 #woordlengte (met kts)
    $teiken="";                                          #def vektor vir teiken
    $wrdsp="";                                           #woord met spasies
    $sp=" ";                                             #spasie
    $s=0;                                                #skuif
    $w="";                                               #nuwe woord sonder -
    @wrdkar = split //, lc($woord1);                     #karakters in 'n array
##########################################################################Kts uit en skep teiken
    for (my $i=0; $i < $wl; $i++){                       #karakters in woord
    if(($wrdkar[$i+$s] =~ /\w|\'|\=/)&
        ($wrdkar[$i+$s+1] =~/\-/)){                      #letter/'/= voor '-'
        $t[$i+$s]=1;                                     #teiken = 1
        $w=$w.$wrdkar[$i+$s];                            #in nuwe wrd (sonder -)
        $s++;                                            #slaan '-' oor
        }
    elsif(($wrdkar[$i+$s] =~ /\w/)&
        ($wrdkar[$i+$s+1] =~/\n/)){                      #laaste letter
        $w=$w.$wrdkar[$i+$s];                            #in nuwe woord
        }
    else{                                                #letter na letter
        $t[$i+$s]=0;                                     #teiken = 0
        $w=$w.$wrdkar[$i+$s];                            #in nuwe woord
        }
    }
    @wrdsonder = split //, $w;
    $g = $#wrdsonder-1;                                  #totale geleenthede
    $totg = $totg + $g;
    $wrdsp=($sp x ($vl-1)).$w.($sp x ($vr-1));           #spasies in woord
    $lwrdsp = length($wrdsp);                            #woord met spasies
    $BBwrdmet="";                                        #lgverdeling met BB
    $laastelet = $wrdsonder[$#wrdsonder];
```

```
#####################################################################Gen vensters & enkodeer
    for(my $j=0; $j < $lwrdsp-$vt+1; $j++){                          #begin VENSTERS
       $venster=substr($wrdsp, $j, $vt);                             #genereer vensters
       @kar=split //, $venster;                                      #karakters in array
       $venkode="";                                                  #gekodeerde invoer
       $kode="";
       for (my $k=0; $k < $vt; $k++){                                #elke karakter in venster
            [ENKODERING soos op bladsy 168]
            $venkode=$venkode.$kode;
            }
    if ($vt == 8){
        $L1 = substr($venkode,0,2);   $L2 = substr($venkode,3,2);
        $L3 = substr($venkode,6,2);   $L4 = substr($venkode,9,2);
        $L5 = substr($venkode,12,2);  $L6 = substr($venkode,15,2);
        $L7 = substr($venkode,18,2);  $L8 = substr($venkode,21,2);
        }
    elsif ($vt == 10){
        $L1 = substr($venkode,0,2);   $L2 = substr($venkode,3,2);
        $L3 = substr($venkode,6,2);   $L4 = substr($venkode,9,2);
        $L5 = substr($venkode,12,2);  $L6 = substr($venkode,15,2);
        $L7 = substr($venkode,18,2);  $L8 = substr($venkode,21,2);
        $L9 = substr($venkode,24,2);  $L10 = substr($venkode,27,2);
        }
    elsif ($vt == 12){
        $L1 = substr($venkode,0,2);   $L2 = substr($venkode,3,2);
        $L3 = substr($venkode,6,2);   $L4 = substr($venkode,9,2);
        $L5 = substr($venkode,12,2);  $L6 = substr($venkode,15,2);
        $L7 = substr($venkode,18,2);  $L8 = substr($venkode,21,2);
        $L9 = substr($venkode,24,2);  $L10 = substr($venkode,27,2);
        $L11 = substr($venkode,30,2); $L12 = substr($venkode,33,2);
        }
    do "LGAfrLet55RulesReg.pl" or die "Reels!";
    if ($class == 1){
        $BBwrdmet = $BBwrdmet.$wrdsonder[$j]."-";
        }
    elsif($class == 0){
        $BBwrdmet = $BBwrdmet.$wrdsonder[$j];
        }
        $telBBuitv++;
    }                                                                #Einde vensters
#############################################################################################
    $BBwrdmet = $BBwrdmet.$laastelet;                                #Laaste letter
    print BBUIT "$BBwrdmet";
    $wl1=length($woord1);
    $wl2=length($BBwrdmet);
    @w1k = split //, $woord1;                                        #woord1 karakters
    @w2k = split //, $BBwrdmet;                                      #woord2 karakters
    $BBs=0;                                                          #Skuifregister
    $BBtots=0;                                                       #Totale skuif in woord
    $BBSIG="";                                                       #Signatuur
    $BBtelSF=0;                                                      #tel foute
    $BBtelGF=0;                                                      #tel misse
```

```
    $BBNW="";                                                    #nuwe woord
    $telBBuit = 0;
################################################################################Tel geleenthede
    for (my $i=0; $i < $wl1; $i++){
        if((($w1k[$i]=~/\w/) & ($w2k[$i+$BBs]=~/\w/))|
            (($w1k[$i]=~/\'/) & ($w2k[$i+$BBs]=~/\'/))|
            (($w1k[$i]=~/\=/) & ($w2k[$i+$BBs]=~/\=/))){         #karakters stem ooreen
            $telBBuit++;                                         #BB uitvoere
            $BBNW=$BBNW."$w1k[$i]";
            $BBW=$BBW."$w1k[$i]";
            $BBgreg++;
            }
        elsif(($w1k[$i] eq "-") & ($w2k[$i+$BBs] eq "-")){       #kt reg ingeplaas
            $BBNW=$BBNW."$w1k[$i]";
            $BBW=$BBW."$w1k[$i]"."($UITVL2[$telBBuit])";
            $BBktregin++;
            $BBgreg++;
            }
        elsif($w1k[$i] eq "-" & $w2k[$i+$BBs] =~ /\w/){          #kt weggelaat
            $BBNW=$BBNW."!";                                     #! waar - moes wees
            $BBW=$BBW."!"."($UITVL2[$telBBuit])";
            $BBwgreg--;
            $BBs--;                                              #vorige kar in wrd2
            $BBtots++;                                           #totale skuif
            $BBgmis++;                                           #geleentheid gemis
            $BBgverkeerd++;                                      #geleentheid verkeerd
            }
        elsif($w1k[$i] =~ /\w/ & $w2k[$i+$BBs] eq "-"){          #kt fout
            $BBNW=$BBNW."*$w1k[$i]";
            $BBW=$BBW."*"."($UITVL2[$telBBuit])"."$w1k[$i]";
            $BBs++;                                              #vlg kar in wrd2
            $BBwgreg--;
            $BBtots++;
            $BBgfout++;                                          #geleentheid fout
            $BBgverkeerd++;                                      #geleentheid verkeerd
            $telBBuit++;
            }
        }
################################################################################################Tel woorde
    if(($BBtots == 0)&($woord1 =~ /\-+/)){
        $BBwreg++;                                               #Woord reg
        $BBwregin++;                                             #sg-woord reg
        print BBREGIN "$BBNW\n";
        print BBWRDES "$BBW\n";
        }
    elsif(($BBtots == 0)&($woord1 !~ /\-/)){
        $BBwreg++;                                               #woord reg
        $BBwregweg++;                                            #ev-woord reg
        print BBREGWEG "$BBNW\n";
        print BBWRDES "$BBW\n";
        }
    elsif($BBtots == -$BBs){
```

```
        $BBwmis++;                                                  #woord met kt(s) gemis
        $BBwverkeerd++;                                             #woord verkeerd
        print BBMIS "$BBNW\n";
        print BBWRDES "$BBW\n";
        }
    else{
        $BBwfout++;                                                 #woord met fout
        $BBwverkeerd++;                                             #Woord verkeerd
        print BBFOUT "$BBNW\n";
        if($BBNW =~ /\*[$C]\-|\-[$C]\*/){
            print BBWKMK "$BBNW\t\t$BBW\n";
            }
        }
    }                                                               #einde woord1
$BBgreg=$totg - $BBgverkeerd;
$BBktregweg = $BBgreg - $BBktregin;
################################################################################Persentasies
$BBpwreg = sprintf("%.2f",$BBwreg/$n*100);                          #% woorde reg
$BBpwverkeerd = sprintf("%.2f",$BBwverkeerd/$n*100);                #% woorde verkeerd
$BBpwregin = sprintf("%.2f",$BBwregin/$n*100);                      #% wrde reg verdeel
$BBpwregweg = sprintf("%.2f",$BBwregweg/$n*100);                    #% woorde reg
$BBpwmis  = sprintf("%.2f",$BBwmis/$n*100);                         #% woorde met misse
$BBpwfout  = sprintf("%.2f",$BBwfout/$n*100);                       #% woorde met foute

$BBpgreg = sprintf("%.2f",$BBgreg/$totg*100);                       #% geleenthede reg
$BBpgverkeerd = sprintf("%.2f",$BBgverkeerd/$totg*100);             #% geleenthede verkeerd
$BBpktregin = sprintf("%.2f",$BBktregin/$totg*100);                 #% kts reg ingeplaas
$BBpktregweg = sprintf("%.2f",$BBktregweg/$totg*100);               #% kts reg weggelaat
$BBpgmis  = sprintf("%.2f",$BBgmis/$totg*100);                      #% geleenthede gemis
$BBpgfout  = sprintf("%.2f",$BBgfout/$totg*100);                    #% geleentheidsfoute

$AKKw = sprintf("%.4f",$BBwreg/($BBwreg+$BBwverkeerd));             #akkuraatheid woordvlak
$PRESw = sprintf("%.4f",$BBwregin/($BBwregin+$BBwfout));            #presisie woordvlak
$HERRw = sprintf("%.4f",$BBwregin/($BBwregin+$BBwmis));             #herroeping woorvlak
$Fw = sprintf("%.2f",2*($PRESw*$HERRw)/($PRESw+$HERRw));            #F-telling woordvlak

$AKKg = sprintf("%.4f",$BBgreg/($BBgreg+$BBgverkeerd));             #akkuraatheid gel-vlak
$PRESg = sprintf("%.4f",$BBktregin/($BBktregin+$BBgfout));          #presisie gel-vlak
$HERRg = sprintf("%.4f",$BBktregin/($BBktregin+$BBgmis));           #herroeping gel-vlak
$Fg = sprintf("%.2f",2*($PRESg*$HERRg)/($PRESg+$HERRg));            #F-telling gel-vlak

print STAT  "\n\n
\t\t\t\t\tBESLISSINGSBOOM \n\n
WOORDVLAK\n
Totale aantal woorde:\t\t\t$n\n
Aantal woorde reg:\t\t\t $BBwreg \t $BBpwreg%
Aantal woorde verkeerd:\t\t\t $BBwverkeerd \t $BBpwverkeerd%\n
Aantal saamgestelde woorde reg:\t\t $BBwregin \t $BBpwregin%
Aantal enkelvoudige woorde reg:\t\t $BBwregweg \t $BBpwregweg%
Aantal woorde met lg gemis:\t\t $BBwmis \t $BBpwmis%
Aantal woorde met foute:\t\t $BBwfout \t $BBpwfout% \n
Akkuraatheid:\t\t$AKKw\n
```

```
Presisie:\t\t$PRESw\n
Herroeping:\t\t$HERRw\n
F-telling:\t\t$Fw\n\n

VERDELINGSGELEENTHEIDSVLAK\n
Totale aantal verdelingsgeleenthede:\t $totg\n
Aantal geleenthede reg:\t\t\t $BBgreg \t $BBpgreg%
Aantal geleenthede verkeerd:\t\t $BBgverkeerd \t $BBpgverkeerd%\n
Aantal koppeltekens reg ingeplaas:\t $BBktregin\t$BBpktregin%
Aantal koppeltekens reg weggelaat:\t $BBktregweg\t$BBpktregweg%
Aantal verdelingsgeleenthede gemis:\t $BBgmis \t $BBpgmis%
Aantal verdelingsgeleentheidsfoute:\t $BBgfout \t $BBpgfout% \n
Akkuraatheid:\t\t$AKKg\n
Presisie:\t\t$PRESg\n
Herroeping:\t\t$HERRg\n
F-telling:\t\t$Fg\n\n
";
close(BBUIT);
close(BBWRDES);
close(BBREG);
close(BBMIS);
close(BBFOUT);
close(BBWKMK);
close(BBFS);
close(STAT);
close(DATA);
```

D.5 Die TEX-algotime

D.5.1 ToetsOP.pl

Dit gebruik OPATGEN-patrone om lettergreepverdeling te doen, prestasie te toets en statistiek te lewer.

```
#!/usr/bin/perl
use utf8;
use TeX::Hyphen;

open(DATA,     "<:utf8", "LG_Toets.txt")  or die "INVOER!!";                  #invoer
open(OPUIT,    ">:utf8", "LGToetsOPUit.txt") or die "OPUitvoer!!";            #wrde volgens OP
open(OPREGWEG, ">:utf8", "ToetsOPREGWEG.txt");                                #ev-wrde reg
open(OPREGIN,  ">:utf8", "ToetsOPREGIN.txt");                                 #sg-woorde reg
open(OPMIS,    ">:utf8", "ToetsOPMIS.txt");                                   #kt mis
open(OPFOUT,   ">:utf8", "ToetsOPFOUT.txt");                                  #kt fout
open(OPWKMK,   ">:utf8", "ToetsOPWKMK.txt");                                  #kts weerskante
open(STAT,     ">:utf8", "ToetsOPSTAT.txt");                                  #statistiek
my $C = "bcdfghjklmnpqrstvwxz";
$woord1= "";
$n=0;                                                                         #woorde
$g=0;                                                                         #geleenthede
```

```
$totg = 0;
$OPwreg=0;                                                          #woorde reg
$OPwverkeerd=0;                                                     #woorde verkeerd
$OPwregin = 0;                                                      #Woord kt reg in
$OPwregweg=0;                                                       #Woord kt reg weg
$OPwmis=0;                                                          #woorde misse
$OPwfout=0;                                                         #woorde foute

$OPgreg=0;                                                          #gel reg
$OPgverkeerd;                                                       #gel verkeerd
$OPktregin = 0;#OP Kt reg ingeplaas
$OPktregweg=0;#OP Kt reg weggelaat
$OPgverkeerd=0;
$OPgfout=0;                                                         #gel fout
$OPgmis=0;#OP geleenthede gemis
$telOPuitv=0;

while($woord1 = <DATA>){                                            #elke woord
    $n++;
##########################################################################################OPATGEN
    $w1 = trim($woord1);
    $w1 =~ s/-//gi;
    $hyp = new TeX::Hyphen 'file' => 'AFRIGpat.tex', style' => 'czech',
    leftmin => 1, rightmin => 1;                                    #patrone
    my @points = $hyp->hyphenate($w1);
    $OPuit = $hyp->visualize($w1);
    print OPUIT "$OPuit\n";
#########################################################################################################
    $w1l = length($w1);
    $w1g = $w1l - 1;
    $totg = $totg + $w1g;
    $wl1=length($woord1);
    $wl2=length($OPuit);
    @w1k = split //, $woord1;                                       #woord1 karakters
    @w2k = split //, $OPuit;                                        #woord2 karakters
    $OPs=0;                                                         #skuifregister
    $OPtots=0;                                                      #totale skuif
    $OPSIG="";                                                      #signatuur
    $OPtelSF=0;                                                     #Tel vir foute
    $OPtelGF=0;                                                     #Teller vir misse
    $OPNW="";                                                       #Woord opgebou
    $telOPuit = 0;
#########################################################################################Tel geleenthede
    for (my $i=0; $i < $wl1; $i++){
        if((($w1k[$i]=~/\w/) & ($w2k[$i+$OPs]=~/\w/))|
            (($w1k[$i]=~/\'/) & ($w2k[$i+$OPs]=~/\'/))|
            (($w1k[$i]=~/\=/) & ($w2k[$i+$OPs]=~/\=/))){            #karakters ooreen
            $telOPuit++;
            $OPNW=$OPNW."$w1k[$i]";
            $OPW=$OPW."$w1k[$i]";
            $OPgreg++;
            }
```

```
        elsif(($w1k[$i] eq "-") & ($w2k[$i+$OPs] eq "-")){                  #kt reg
            $OPNW=$OPNW."$w1k[$i]";
            $OPktregin++;
            $OPgreg++;
            }
        elsif($w1k[$i] eq "-" & $w2k[$i+$OPs] =~ /\w/){                     #kt gemis
            $OPNW=$OPNW."!";                                                #! by mis
            $OPgreg--;
            $OPs--;                                                         #vorige kar wrd2
            $OPtots++;                                                      #totale skuif
            $OPgmis++;                                                      #gel gemis
            $OPgverkeerd++;                                                 #gel verkeerd
            }
        elsif($w1k[$i] =~ /\w/ & $w2k[$i+$OPs] eq "-"){                     #kt fout
            $OPNW=$OPNW."*$w1k[$i]";
            $OPs++;                                                         #vlg kar wrd2
            $OPgreg--;
            $OPtots++;
            $OPgfout++;                                                     #gel fout
            $OPgverkeerd++;
            $telOPuit++;
            }
        }
##############################################################################################Tel woorde
    if(($OPtots == 0)&($woord1 =~ /\-+/)){
        $OPwreg++;                                                          #woord reg
        $OPwregin++;                                                        #sg woord reg
        print OPREGIN "$NNNW\n";
        }
    elsif(($OPtots == 0)&($woord1 !~ /\-/)){
        $OPwreg++;
        $OPwregweg++;                                                       #ev woord reg
        print OPREGWEG "$OPNW\n";
        }
    elsif($OPtots == -$OPs){
        $OPwmis++;                                                          #woord met mis
        $OPwverkeerd++;                                                     #woord verkeerd
        print OPMIS "$OPNW\n";
        }
    else{
        $OPwfout++;                                                         #woord met fout
        $OPwverkeerd++;
        print OPFOUT "$OPNW\n";
        if($OPNW =~ /\*[$C]\-|\-[$C]\*/){
            print OPWKMK "$OPNW\n";
            }
        }
    }                                                                       #Einde woorde
$OPgreg=$totg - $OPgverkeerd;
$OPktregweg = $OPgreg - $OPktregin;
##############################################################################################Persentasies
$OPpwreg = sprintf("%.2f",$OPwreg/$n*100);                                  #% wrde reg
```

```
$OPpwverkeerd = sprintf("%.2f",$OPwverkeerd/$n*100);                    #% wrde verkeerd
$OPpwregin = sprintf("%.2f",$OPwregin/$n*100);                          #% wrde reg verdeel
$OPpwregweg = sprintf("%.2f",$OPwregweg/$n*100);                        #% wrde reg
$OPpwmis  = sprintf("%.2f",$OPwmis/$n*100);                             #% wrde met misse
$OPpwfout  = sprintf("%.2f",$OPwfout/$n*100);                           #% wrde met foute

$OPpgreg = sprintf("%.2f",$OPgreg/$totg*100);                           #% gel reg
$OPpgverkeerd = sprintf("%.2f",$OPgverkeerd/$totg*100);                 #% kts verkeerd
$OPpktregin = sprintf("%.2f",$OPktregin/$totg*100);                     #% kts reg in
$OPpktregweg = sprintf("%.2f",$OPktregweg/$totg*100);                   #% kts reg weg
$OPpgmis  = sprintf("%.2f",$OPgmis/$totg*100);                          #% gel gemis
$OPpgfout  = sprintf("%.2f",$OPgfout/$totg*100);                        #% gel foute

print STAT  "\n\n
\t\t\t\t\tOPATGEN \n\n
Totale aantal woorde:\t\t\t$n\n\n
Aantal woorde reg:\t\t\t $OPwreg \t $OPpwreg%
Aantal woorde verkeerd:\t\t\t $OPwverkeerd \t $OPpwverkeerd%\n
Aantal saamgestelde woorde reg:\t\t $OPwregin \t $OPpwregin%
Aantal enkelvoudige woorde reg:\t\t $OPwregweg \t $OPpwregweg%
Aantal woorde met lg gemis:\t\t $OPwmis \t $OPpwmis%
Aantal woorde met foute:\t\t $OPwfout \t $OPpwfout% \n\n

Totale aantal verdelingsgeleenthede:\t $totg\n
Aantal geleenthede reg:\t\t\t $OPgreg \t $OPpgreg%
Aantal geleenthede verkeerd:\t\t $OPgverkeerd \t $OPpgverkeerd%\n
Aantal koppeltekens reg ingeplaas:\t $OPktregin\t$OPpktregin%
Aantal koppeltekens reg weggelaat:\t $OPktregweg\t$OPpktregweg%
Aantal verdelingsgeleenthede gemis:\t $OPgmis \t $OPpgmis%
Aantal verdelingsgeleentheidsfoute:\t $OPgfout \t $OPpgfout% ";
close(OPUIT);
close(NNWRDES);
close(OPREGWEG);
close(OPREGIN);
close(OPMIS);
close(OPFOUT);
close(OPWKMK);
close(STAT);
close(DATA);
```

D.6 K-Algoritme

D.6.1 VergelykMetLGLys.pl

Vergelyk woorde uit 'n bron met ons leksikon en lewer woorde wat in beide lyste voorkom (InBeide) en woorde wat net in die bron voorkom (NetHier).

```
#!/usr/bin/perl
use utf8;
```

```
require List::Compare;

open(LYS,       "<:utf8",   "JohUniek.txt")    or die ("Invoer1!!\n");
open(LG_LYS,    "<:utf8",   "LG_Lys.txt")  or die ("Invoer2!!\n");
open(INBEIDE,   ">:utf8",   "JohInBeide.txt")  or die ("Uitvoer1!!\n");
open(UNIEK,    ">:utf8",   "JohNetHier.txt")  or die ("Uitvoer2!!\n");
while ($wrd = <LG_LYS>){                                            #woord met kts
        $wrd =~ s/-//gi;                                            #verwyder kts
        push @LGLYS, $wrd;
        }
while ($w = <LYS>){
        push @LYS,$w;
        }
$lca = List::Compare->new('-a', \@LYS, \@LGLYS);
@intersection = $lca->get_intersection;
@unique = $lca->get_unique;
print INBEIDE "@intersection";
print UNIEK "@unique";

close(LYS);
close(LG_LYS);
close(INBEIDE);
close(UNIEK);
```

D.6.2 VerdWrdUitLGLys.pl

Onttrek lettergreepverdeelde woorde vir die woorde in InBeide uit die leksikon wat in lettergrepe verdeel is (LG_Lys.txt).

```
#!/usr/bin/perl
use utf8;

open(INBEIDE,   "<:utf8",   "JohInBeide.txt")        or die ("Invoer1!\n");
open(LGLYS,     "<:utf8",   "LG_Lys.txt")            or die ("Invoer2!\n");
open(UITVOER,   ">:utf8",   "JohInBeideVerdeel.txt") or die ("Uitvoer1!!\n");
while ($line = <LGLYS>){
    push @LGL, $line;                                                   #woord met kts
    $line =~ s/-//g;                                                    #woord sonder kts
    push @LGLSonder, $line;
    }
while ($w = <INBEIDE>){
        for ($i = 0; $i <= $#LGLSonder; $i++){
                if ($w eq $LGLSonder[$i]){
                        print UITVOER $LGL[$i];
                        }
                }
        }
close(INBEIDE);
close(LGLYS);
close(UITVOER);
```

D.6.3 ALG5_LGToets.pl

Verdeel woorde wat in beide lyste voorkom met ALG5 en toets dit teen die lettergreepverdeelde woorde.

```
#!/usr/bin/perl
#OPSG -> OPLG + NNLG

use utf8;
use TeX::Hyphen;

open(DATA,                 "<:utf8", "ToetsLys.txt");                #lg-verdeelde wrde
open(UIT,                  ">:utf8", "ToetsLys_UIT_ALG5.txt");       #NN se Lg-verdeling
open(InvoerGewigte_LG,     "<IWLGLys53160.txt") or die "Inv Gew!";
open(VerborgeBeladings_LG, "<HBLGLys53160.txt") or die "Verb Bel!";
open(VerborgeGewigte_LG,   "<HWLGLys53160.txt") or die "Verb Gew!";
open(UitvoerBeladings_LG,  "<OBLGLys53160.txt") or die "Uitv Bel!";

open(REGWEG,    ">:utf8", "ToetsLys_REGWEG_ALG5.txt");              #Kt reg weggelaat
open(REGIN,     ">:utf8", "ToetsLys_REGIN_ALG5.txt");               #Kt reg ingeplaas
open(MIS,       ">:utf8", "ToetsLys_MIS_ALG5.txt");                 #Kt gemis
open(FOUT,      ">:utf8", "ToetsLys_FOUT_ALG5.txt");                #Kt verkeerd
open(STAT,      ">:utf8", "ToetsLys_STAT_ALG5.txt");                #Statistiek

while ($VB_LG = <VerborgeBeladings_LG>){
    $VerbB_LG = trim($VB_LG);
    push(@VerbB_LG, $VerbB_LG);
    }
while ($UitvB_LG = <UitvoerBeladings_LG>){
    $UB_LG = trim($UitvB_LG);
    push @UitvB_LG, $UB_LG;
    }
while ($line = <VerborgeGewigte_LG>){
    $VG_LG = trim($line);
    @VerbG_LG = split(/\s+/, $VG_LG);                               #verdeel by witspasies
    }
while (<InvoerGewigte_LG>){
    push @InvG_LG, [ split ];
    }
my $C = "bcdfghjklmnpqrstvwxz";
$WRD= "";
$n=0;                                                               #woorde tel
$totg=0;
$WRDreg=0;                                                          #woorde reg
$WRDverkeerd=0;                                                     #woorde verkeerd
$WRDregin=0;                                                        #woord met kt(s) reg ingeplaas
$WRDregweg=0;                                                       #Woord met kt(s) reg weggelaat
$WRDmis=0;                                                          #woorde met kts gemis
$WRDfout=0;                                                         #woorde met foute (en misse)
$TOTgreg=0;                                                         #verdelingsgeleenthede reg
$TOTgverkeerd=0;                                                    #verdelingsgeleenthede verkeerd
```

```
$TOTgregin=0;                                                    #kt reg ingeplaas
$TOTgregweg=0;                                                   #kt reg weggelaat
$TOTgfout=0;                                                     #geleenthede fout
$TOTgmis=0;                                                      #geleenthede gemis
while($WRD = <DATA>){                                            #vir elke woord in invoer
    $g=0;                                                        #geleenthede per wrd tel
    $n++;
    print "$n\n";
    $VenLNN=5;  $VenRNN=3;                                       #venster links en regs
    $VenTotNN=$VenLNN+$VenRNN;                                   #totale venster
    $WSonder = trim($WRD);
    $WSonder =~ s/-//gi;
    @WSonder = split //, $WSonder;
    $laastelet = $WSonder[$#WSonder];
    OPSG($WSonder);
    $SGW = $OPSG;
    @SGW = split //, $SGW;
    OPLG($SGW);
    NNLG($SGW);
    if ($OPLG eq $NNLG){
        $LGW = $OPLG;
        $LGW =~ s/--/-/;
        $LGW =~ s/-=/=/;
        if ($LGW =~ /-[$C][$C]$/){
               $mks = substr($LGW, -3);
               $LGW =~ s/-[$C][$C]$/$mks/;
               chomp $LGW;
               }
        print UIT "$LGW";
        }
    else{
        $LGW = "";
        $s = 0;
        for (my $i=0; $i <= $LOPLG; $i++){
                if ($OPLG[$i] eq $NNLG[$i+$s]){
                       $LGW = $LGW."$OPLG[$i]";
                       }
                elsif(($OPLG[$i] eq "-") & ($NNLG[$i+$s] =~ /\w/)){
                       $s--;
                       }
               elsif(($OPLG[$i] =~ /\w/) & ($NNLG[$i+$s] eq "-")){
                       $s++;
                       $LGW = $LGW."$OPLG[$i]";
                       }
               }
        $LGW =~ s/--/-/;
        $LGW =~ s/-=/=/;
        $LGW =~ s/-[$C]$/[$C]/;
        if ($LGW =~ /-[$C][$C]$/){
        $mks = substr($LGW, -3);
                $LGW =~ s/-[$C][$C]$/$mks/;
                chomp $LGW;
```

```
                }
        print UIT "$LGW";
        }
################################################################################Ontleed uitvoer
    $wg = $#WSonder-1;
    $totg = $totg + $wg;
    $TOTgreg = $TOTgreg + $wg;
    $wgreg=$wg;
    $WRDreg++;
    $wgverkeerd=0;
    $wgregin=0;
    $wgregweg=0;
    $wgfout=0;
    $wgmis=0;
    $LWRD=length($WRD);
    $LLGW=length($LGW);
    @LGW = split //, $LGW;
    @WRD = split //, $WRD;
    $s = 0;
    $tots = 0;
    for (my $i=0; $i < $LWRD; $i++){
        if(($WRD[$i] eq "-") & ($LGW[$i+$s] eq "-")){
               $TOTgregin++;                                    #kt reg ingeplaas
               $wgregin++;
               }
        elsif($WRD[$i] eq "-" & $LGW[$i+$s] =~ /\w/){           #kt weggelaat - mis
            $s--;                                               #gebruik vorige kar in wrd2
            $tots++;
            $TOTgmis++;                                         #geleentheid gemis
                $wgmis++;
                $TOTgverkeerd++;                                #geleentheid verkeerd
                $wgverkeerd++;
                $wgreg--;
                $TOTgreg--;
                }
        elsif($WRD[$i] =~ /\w/ & $LGW[$i+$s] eq "-"){           #kt verkeerd ingevoeg - fout
               $s++;                                            #gebruik volgende kar in wrd2
               $tots++;
               $TOTgfout++;                                     #geleentheid fout
               $wgfout++;
               $TOTgverkeerd++;                                 #geleentheid verkeerd
               $wgverkeerd++;
               $wgreg--;
               $TOTgreg--;
               }
        }
        $wgregweg = $wg - $wgregin-$wgmis-$wgfout;
        $TOTgregweg = $TOTgregweg + $wgregweg;
        if (($tots == 0)&($WRD =~ /\-+/)){
               $WRDregin++;                                     #SG woord reg
               print REGIN "$LGW";
               }
```

```
        elsif(($tots == 0)&&($WRD !~ /\-/)){
               $WRDregweg++;                                        #EV woord reg
               print REGWEG "$LGW";
               }
        elsif($tots == -$s){
               $WRDmis++;                                           #Woord met kt(s) gemis
               $WRDverkeerd++;                                      #Woord verkeerd
               $WRDreg--;
               print MIS "$LGW";
               }
        else{
                $WRDfout++;                                         #Woord met kts verkeerd
                $WRDverkeerd++;                                     #Woord verkeerd
                $WRDreg--;
                print FOUT "$LGW";
                }
        }
$pWRDreg = sprintf("%.2f",$WRDreg/$n*100);                          #% woorde reg
$pWRDverkeerd = sprintf("%.2f",$WRDverkeerd/$n*100);                #% woorde verkeerd
$pWRDregin = sprintf("%.2f",$WRDregin/$n*100);                      #% sg-wrde reg verdeel
$pWRDregweg = sprintf("%.2f",$WRDregweg/$n*100);                    #% ev-woorde reg
$pWRDmis  = sprintf("%.2f",$WRDmis/$n*100);                         #% woorde met goeie foute
$pWRDfout  = sprintf("%.2f",$WRDfout/$n*100);                       #% woorde met slegte foute

$pTOTgreg = sprintf("%.2f",$TOTgreg/$totg*100);                     #% koppeltekens reg ingeplaas
$pTOTgverkeerd = sprintf("%.2f",$TOTgverkeerd/$totg*100);           #% geleenthede verkeerd
$pTOTgregin = sprintf("%.2f",$TOTgregin/$totg*100);                 #% kts reg ingeplaas
$pTOTgregweg = sprintf("%.2f",$TOTgregweg/$totg*100);               #% kts reg weggelaat
$pTOTgmis  = sprintf("%.2f",$TOTgmis/$totg*100);                    #% geleenthede gemis
$pTOTgfout  = sprintf("%.2f",$TOTgfout/$totg*100);                  #% geleentheidsfoute

$AW  = sprintf("%.2f",($WRDregin + $WRDregweg)/($WRDregin+$WRDregweg+$WRDfout+$WRDmis)*100);
$PW = sprintf("%.2f",$WRDregin/($WRDregin+$WRDfout)*100);
$RW = sprintf("%.2f",$WRDregin/($WRDregin+$WRDmis)*100);
$FW = sprintf("%.2f",2*($PW*$RW)/($PW+$RW));

$AG  = sprintf("%.2f",($TOTgregin + $TOTgregweg)/($TOTgregin+$TOTgregweg+$TOTgfout+$TOTgmis)*100);
$PG = sprintf("%.2f",$TOTgregin/($TOTgregin+$TOTgfout)*100);
$RG = sprintf("%.2f",$TOTgregin/($TOTgregin+$TOTgmis)*100);
$FG = sprintf("%.2f",2*($PG*$RG)/($PG+$RG));

print STAT  "\n\n
\t\t\t\t\tSTATISTIEK\n\n
Totale aantal woorde:                   $n\n\n
Woorde reg:                             $WRDreg         $pWRDreg%
Woorde verkeerd:                        $WRDverkeerd    $pWRDverkeerd%\n
Saamgestelde woorde reg:                $WRDregin       $pWRDregin%
Enkelvoudige woorde reg:                $WRDregweg      $pWRDregweg%
Woorde met lg gemis:                    $WRDmis         $pWRDmis%
Woorde met foute:                       $WRDfout        $pWRDfout% \n\n

Totale aantal verdelingsgeleenthede:    $totg\n\n
```

```
Totale verdelingsgeleenthede reg:       $TOTgreg       $pTOTgreg%
Totale verdelingsgeleenthede verkeerd:  $TOTgverkeerd  $pTOTgverkeerd%\n
Koppeltekens reg ingeplaas:             $TOTgregin     $pTOTgregin%
Koppeltekens reg weggelaat:             $TOTgregweg    $pTOTgregweg%
Verdelingsgeleenthede gemis:            $TOTgmis       $pTOTgmis%
Verdelingsgeleentheidsfoute:            $TOTgfout      $pTOTgfout%\n\n

Woordvlak       Geleentheidsvlak\n
Akkuraatheid:    $AW%                $AG%\n
Presisie:        $PW%                $PG%\n
Herroep:         $RW%                $RG%\n
F-statistiek:    $FW%                $FG%   ";

close(DATA);
close(UIT);
close(InvoerGewigte_LG);
close(VerborgeBeladings_LG);
close(VerborgeGewigte_LG);
close(UitvoerBeladings_LG);
close(InvoerGewigte_SG);
close(VerborgeBeladings_SG);
close(VerborgeGewigte_SG);
close(UitvoerBeladings_SG);
close(REGWEG);
close(REGIN);
close(MIS);
close(FOUT);
close(STAT);
##############################################################################################OPLG
sub OPLG{
    $hyp = new TeX::Hyphen  'file' => 'LGPatVoll.tex', 'style' => 'german',
           leftmin => 1, rightmin => 1;                              #PATRONE
    my @points = $hyp->hyphenate($SGW);
    $OPLG = $hyp->visualize($SGW);
    $OPLG =~ s/-$//;
    $OPLG =~ s/--/-/;
    $OPLG =~ s/-=/=/;
    $een = substr($OPLG,-2);
    $OPLG =~ s/\-[bcdfghjklmnpqrstvwxz]$/$een/;
    @OPLG = split //, $OPLG;
    $LOPLG = length($OPLG);
    }
##############################################################################################OPSG
sub OPSG{
    $OPSG = "";
    @OPSG = "";
    $hyp = new TeX::Hyphen  'file' => 'SGWPatVoll.tex', 'style' => 'german',
           leftmin => 1, rightmin => 1;                              #PATRONE
    my @points = $hyp->hyphenate($WSonder);
    $OPSG = $hyp->visualize($WSonder);
    $OPSG =~ s/-$//;
    $OPSG =~ s/--/-/;
```

```
    $OPSG =~ s/-=/=/;
    $OPL = length($OPSG);
    @OPSG = split //, $OPSG;
    }
##############################################################################################NNLG
sub NNLG{
    @UITVL2="";
    $wrdsp="";                                                      #woord met spasies
    $sp=" ";                                                        #spasie
    $w="";                                                          #nuwe woord sonder kt
    $wrdsp=($sp x ($VenLNN-1)).$SGW.($sp x ($VenRNN-1));            #plaas spasies in
    $lwrdsp = length($wrdsp);                                       #lengte van woord met sp
    $NNwrdmet="";                                                   #Afbreking met NN
#########################################################################GENEREER VENSTERS EN KODEER
   for(my $j=0; $j <= $lwrdsp-$VenTotNN; $j++){                     #VENSTERS: skuif oor $wrdsp
       $venster=substr($wrdsp, $j, $VenTotNN);                      #vensters van lengte vt
       @kar=split //, $venster;                                     #vensterkarakters in array
       $venkode="";                                                 #gekodeerde invoer
       $kode="";
       for (my $k=0; $k < $vt; $k++){                               #elke karakter in venster
           [ENKODERING SOOS OP BLADSY 166]
           }
           $venkode=$venkode.$kode;                                 #invoerry vir venster
           }
##############################################################################################NN uitvoer
    @Invoer = split(/\s+/, $venkode);
    for $c (0 .. $#InvG_LG){                                        #Verb neurone 1 vir 1
         $Som1[$c] = 0;
         for $d (0 .. $#{$InvG_LG[$c]}){                            #InvGewigte na neuron c
              $InvL1[$d]=$Invoer[$d]*$InvG_LG[$c][$d];              #Ven-element X InvG
              $Som1[$c] = $Som1[$c]+$InvL1[$d];                     #Tel bymekaar
              }
         $Net1[$c]=$Som1[$c] + $VerbB_LG[$c];
         $UitvL1[$c] = 1/(1+exp(-($Net1[$c])));                     #Uitvoer1:logsig oordragf
         }
    $Som2 = 0;
    for $e (0 .. $#VerbG_LG){                                       #elke verborge gewig
         $InvL2[$e] = $UitvL1[$e] * $VerbG_LG[$e];                  #UitvL1 x VerbG.
         $Som2 = $Som2 + $InvL2[$e];
         }
    $Net2 = $Som2 + $UitvB_LG[0];
    $UitvL2 = 1/(1+exp(-($Net2)));                                  #Uitvoer2: logsig oordragf
    if (($UitvL2 < 0.5)|($SGW[$j] eq "-") ){
         $NNwrdmet = $NNwrdmet.$SGW[$j];
         }
    else{
         $NNwrdmet = $NNwrdmet.$SGW[$j]."-";
         }
    }                                                               #Einde vensters
    NNLG = $NNwrdmet.$laastelet;                                    #Laaste letter van woord
    $NNLG =~ s/-$//;
    $NNLG =~ s/--/-/;
```

```
    $NNLG =~ s/--/=/;
    @NNLG = split //, $NNLG;
    $LNNLG = length($NNLG);
    }
close(NNUIT);
close(NNWRDES);
close(NNREG);
close(NNMIS);
close(NNFOUT);
close(NNWKMK);
close(NNFS);
close(STAT);
close(DATA);
close(NNUIT);
close(InvoerGewigte);
close(VerborgeBeladings);
close(VerborgeGewigte);
close(UitvoerBeladings);
```

D.6.4 ALG5Toepas.pl

Pas ALG5 op woorde wat nie in LG_Lys.txt voorkom nie (NetHier) toe. Die uitvoer word per hand beoordeel.

```
#!/usr/bin/perl
#OPSG -> OPLG + NNLG

use utf8;
use TeX::Hyphen;

($sec,$min,$hour,$mday,$mon,$year,$wday,
$yday,$isdst)=localtime(time);
print "$hour:$min:$sec\n";

open(DATA,                 "<:utf8", "ToetsLys.txt");                #onverdeelde wrde
open(UIT,                  ">:utf8", "ToetsLys_UIT_ALG5.txt");       #NN se lg-verdeling
open(InvoerGewigte_LG,     "<IWLGLys53160.txt") or die "Inv Gew!";
open(VerborgeBeladings_LG, "<HBLGLys53160.txt") or die "Verb Bel!";
open(VerborgeGewigte_LG,   "<HWLGLys53160.txt") or die "Verb Gew!";
open(UitvoerBeladings_LG,  "<OBLGLys53160.txt") or die "Uitv Bel!";

while ($VB_LG = <VerborgeBeladings_LG>){
    $VerbB_LG = trim($VB_LG);
    push(@VerbB_LG, $VerbB_LG);
    }
while ($UitvB_LG = <UitvoerBeladings_LG>){
    $UB_LG = trim($UitvB_LG);
    push @UitvB_LG, $UB_LG;
    }
while ($line = <VerborgeGewigte_LG>){
```

```
    $VG_LG = trim($line);
    @VerbG_LG = split(/\s+/, $VG_LG);                                    #verdeel by witspasies
    }
while (<InvoerGewigte_LG>){
    push @InvG_LG, [ split ];
    }
my $C = "bcdfghjklmnpqrstvwxz";

$WRD= "";
$n=0;                                                                    #woorde tel

while($WRD = <DATA>){                                                    #vir elke woord in invoer
    $g=0;                                                                #geleenthede per wrd tel
    $n++;
    print "$n\n";
    $VenLNN=5; $VenRNN=3;                                                #lg venster links en regs
    $VenTotNN=$VenLNN+$VenRNN;                                           #totale venster
    @WSonder = split //, $WRD;
    $laastelet = $WSonder[$#WSonder];
    OPSG($WRD);
    $SGW = $OPSG;
    @SGW = split //, $SGW;
    OPLG($SGW);
    NNLG($SGW);
    if ($OPLG eq $NNLG){
        $LGW = $OPLG;
        $LGW =~ s/--/-/;
        print UIT "$LGW";
        }
    else{
        $LGW = "";
        $s = 0;
        for (my $i=0; $i <= $LOPLG; $i++){
        if ($OPLG[$i] eq $NNLG[$i+$s]){
                $LGW = $LGW."$OPLG[$i]";
                }
            elsif(($OPLG[$i] eq "-") & ($NNLG[$i+$s] =~ /\w/)){
                $s--;
                }
            elsif(($OPLG[$i] =~ /\w/) & ($NNLG[$i+$s] eq "-")){
                $s++;
                $LGW = $LGW."$OPLG[$i]";
                }
            }
    $LGW =~ s/--/-/;
    print UIT "$LGW";
    }
    if ($LGW =~ /-\w{6,}-|^\w{6,}-|-\w{6,}$/){
        print VLAGGIE "$LGW\n";
        }
    }
close(DATA);
```

```
close(UIT);
close(InvoerGewigte_LG);
close(VerborgeBeladings_LG);
close(VerborgeGewigte_LG);
close(UitvoerBeladings_LG);
close(InvoerGewigte_SG);
close(VerborgeBeladings_SG);
close(VerborgeGewigte_SG);
close(UitvoerBeladings_SG);
#############################################################################################OPLG
sub OPLG{
    $hyp = new TeX::Hyphen  'file' => 'LGPatVoll.tex', 'style' => 'german',
           leftmin => 1, rightmin => 1;                                    #PATRONE!!!
    my @points = $hyp->hyphenate($SGW);
    $OPLG = $hyp->visualize($SGW);
    $OPLG =~ s/-$//;
    $OPLG =~ s/--/-/;
    $OPLG =~ s/-=/=/;
    $een = substr($OPLG,-2);
    $OPLG =~ s/\-[bcdfghjklmnpqrstvwxz]$/$een/;
    @OPLG = split //, $OPLG;
    $LOPLG = length($OPLG);
    }
#############################################################################################OPSG
sub OPSG{
    $OPSG = "";
    @OPSG = "";
    $hyp = new TeX::Hyphen  'file' => 'SGWPatVoll.tex', 'style' => 'german',
           leftmin => 1, rightmin => 1;                                     #PATRONE!!!
    my @points = $hyp->hyphenate($WSonder);
    $OPSG = $hyp->visualize($WSonder);
    $OPSG =~ s/-$//;
    $OPSG =~ s/--/-/;
    $OPSG =~ s/-=/=/;
    $OPL = length($OPSG);
    @OPSG = split //, $OPSG;
    }
#############################################################################################NNLG
sub NNLG{
    @UITVL2="";
    $wrdsp="";                                                          #woord met spasies
    $sp=" ";                                                            #spasie
    $w="";                                                              #nuwe woord sonder kt
    $wrdsp=($sp x ($VenLNN-1)).$SGW.($sp x ($VenRNN-1));                #plaas spasies in
    $lwrdsp = length($wrdsp);                                           #lengte van woord met sp
    $NNwrdmet="";                                                       #Afbreking met NN
#########################################################################GENEREER VENSTERS EN KODEER
   for(my $j=0; $j <= $lwrdsp-$VenTotNN; $j++){                         #VENSTERS: skuif oor $wrdsp
       $venster=substr($wrdsp, $j, $VenTotNN);                          #vensters van lengte vt
       @kar=split //, $venster;                                         #vensterkarakters in array
       $venkode="";                                                     #gekodeerde invoer
       $kode="";
```

```
       for (my $k=0; $k < $VenTotNN; $k++){                          #elke karakter in venster
           [ENKODERING SOOS OP BLADSY 166]
               $venkode=$venkode.$kode;                              #invoerry vir venster
               }
################################################################################NN uitvoer
    @Invoer = split(/\s+/, $venkode);
    for $c (0 .. $#InvG_LG){                                         #Verb neurone 1 vir 1
         $Som1[$c] = 0;
         for $d (0 .. $#{$InvG_LG[$c]}){                             #InvGewigte na neuron c
              $InvL1[$d]=$Invoer[$d]*$InvG_LG[$c][$d];               #Ven-element X InvG
              $Som1[$c] = $Som1[$c]+$InvL1[$d];                      #Tel bymekaar
              }
         $Net1[$c]=$Som1[$c] + $VerbB_LG[$c];
         $UitvL1[$c] = 1/(1+exp(-($Net1[$c])));                      #Uitvoer1: logsig oordragf
         }                                                           #Einde uitvoer vir laag 1
    $Som2 = 0;
    for $e (0 .. $#VerbG_LG){                                        #elke verborge gewig
         $InvL2[$e] = $UitvL1[$e] * $VerbG_LG[$e];                   #UitvL1 x VerbG.
         $Som2 = $Som2 + $InvL2[$e];
         }
    $Net2 = $Som2 + $UitvB_LG[0];
    $UitvL2 = 1/(1+exp(-($Net2)));                                   #Uitvoer2: logsig oordragf
    if (($UitvL2 < 0.5)|($SGW[$j] eq "-") ){
         $NNwrdmet = $NNwrdmet.$SGW[$j];
         }
    else{
         $NNwrdmet = $NNwrdmet.$SGW[$j]."-";
         }
    }                                                                #einde vensters
    NNLG = $NNwrdmet.$laastelet;                                     #laaste letter
    $NNLG =~ s/-$//;
    $NNLG =~ s/--/-/;
    $NNLG =~ s/-=/=/;
    @NNLG = split //, $NNLG;
    $LNNLG = length($NNLG);
    }
close(NNUIT);
close(NNWRDES);
close(NNREG);
close(NNMIS);
close(NNFOUT);
close(NNWKMK);
close(NNFS);
close(STAT);
close(DATA);
close(NNUIT);
close(InvoerGewigte);
close(VerborgeBeladings);
close(VerborgeGewigte);
close(UitvoerBeladings);
```

D.7 Enkodering

D.7.1 Neurale netwerke

```
  if (kar[$k] eq " "){$kode="1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
                            0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 "} #1
elsif(kar[$k] eq "a"){$kode="0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
                            0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 "} #2
elsif(kar[$k] eq "á"){$kode="0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
                            0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 "} #3
elsif(kar[$k] eq "ä"){$kode="0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
                            0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 "} #4
elsif(kar[$k] eq "e"){$kode="0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
                            0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 "} #5
elsif(kar[$k] eq "é"){$kode="0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
                            0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 "} #6
elsif(kar[$k] eq "è"){$kode="0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
                            0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 "} #7
elsif(kar[$k] eq "ë"){$kode="0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
                            0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 "} #8
elsif(kar[$k] eq "ê"){$kode="0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
                            0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 "} #9
elsif(kar[$k] eq "i"){$kode="0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
                            0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 "} #10
elsif(kar[$k] eq "í"){$kode="0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
                            0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 "} #11
elsif(kar[$k] eq "ï"){$kode="0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
                            0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 "} #12
elsif(kar[$k] eq "î"){$kode="0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
                            0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 "} #13
elsif(kar[$k] eq ö"){$kode="0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
                            0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 "} #14
elsif(kar[$k] eq "ó"){$kode="0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
                            0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 "} #15
elsif(kar[$k] eq "ò"){$kode="0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
                            0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 "} #16
elsif(kar[$k] eq "ö"){$kode="0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
                            0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 "} #17
elsif(kar[$k] eq "ô"){$kode="0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
                            0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 "} #18
elsif(kar[$k] eq ü"){$kode="0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
                            0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 "} #19
```

```
elsif(kar[$k] eq "ú"){$kode="0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
                             0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 "} #20
elsif(kar[$k] eq "ü"){$kode="0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
                             0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 "} #21
elsif(kar[$k] eq "û"){$kode="0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
                             0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 "} #22
elsif(kar[$k] eq ij"){$kode="0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
                             0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 "} #23
elsif(kar[$k] eq "ý"){$kode="0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
                             0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 "} #24
elsif(kar[$k] eq "b"){$kode="0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
                             0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 "} #25
elsif(kar[$k] eq "c"){$kode="0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
                             0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 "} #26
elsif(kar[$k] eq "d"){$kode="0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
                             0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 "} #27
elsif(kar[$k] eq "f"){$kode="0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
                             0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 "} #28
elsif(kar[$k] eq "g"){$kode="0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
                             1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 "} #29
elsif(kar[$k] eq "h"){$kode="0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
                             0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 "} #30
elsif(kar[$k] eq "j"){$kode="0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
                             0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 "} #31
elsif(kar[$k] eq "k"){$kode="0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
                             0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 "} #32
elsif(kar[$k] eq "l"){$kode="0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
                             0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 "} #33
elsif(kar[$k] eq "m"){$kode="0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
                             0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 "} #34
elsif(kar[$k] eq "n"){$kode="0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
                             0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 "} #35
elsif(kar[$k] eq "p"){$kode="0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
                             0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 "} #36
elsif(kar[$k] eq "q"){$kode="0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
                             0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 "} #37
elsif(kar[$k] eq "r"){$kode="0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
                             0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 "} #38
elsif(kar[$k] eq "s"){$kode="0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
                             0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 "} #39
elsif(kar[$k] eq "t"){$kode="0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
                             0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 "} #40
elsif(kar[$k] eq "v"){$kode="0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

```
                              0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 "} #41
elsif(kar[$k] eq "w"){$kode="0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
                              0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 "} #42
elsif(kar[$k] eq "x"){$kode="0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
                              0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 "} #43
elsif(kar[$k] eq "z"){$kode="0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
                              0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 "} #44
elsif(kar[$k] eq "="){$kode="0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
                              0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 "} #45
elsif(kar[$k] eq "´"){$kode="0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
                              0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 "} #46
```

D.7.2 Beslissingsbome

```
   if(kar[$k] eq " "){$kode="Sp "} #1    elsif(kar[$k] eq "a"){$kode="Aa "} #2
elsif(kar[$k] eq "á"){$kode="Ak "} #3    elsif(kar[$k] eq "ä"){$kode="Ad "} #4
elsif(kar[$k] eq "e"){$kode="Ee "} #5     elsif(kar[$k] eq "é"){$kode="Ea "} #6
elsif(kar[$k] eq "è"){$kode="Eg "} #7     elsif(kar[$k] eq "ë"){$kode="Ed "} #8
elsif(kar[$k] eq "ê"){$kode="Ek "} #9     elsif(kar[$k] eq "i"){$kode="Ii "} #10
elsif(kar[$k] eq "í"){$kode="Ia "} #11    elsif(kar[$k] eq "ï"){$kode="Id "} #12
elsif(kar[$k] eq "î"){$kode="Ik "} #13    elsif(kar[$k] eq "o"){$kode="Oo "} #14
elsif(kar[$k] eq "ó"){$kode="Oa "} #15    elsif(kar[$k] eq "ò"){$kode="Og "} #16
elsif(kar[$k] eq "ö"){$kode="Od "} #17    elsif(kar[$k] eq "ô"){$kode="Ok "} #18
elsif(kar[$k] eq "u"){$kode="Uu "} #19    elsif(kar[$k] eq "ú"){$kode="Ua "} #20
elsif(kar[$k] eq "ü"){$kode="Ud "} #21    elsif(kar[$k] eq "û"){$kode="Uk "} #22
elsif(kar[$k] eq "y"){$kode=IJy "} #23    elsif(kar[$k] eq "ý"){$kode=IJa "} #24
elsif(kar[$k] eq "b"){$kode="Bb "} #25    elsif(kar[$k] eq "c"){$kode="Cc "} #26
elsif(kar[$k] eq "d"){$kode="Dd "} #27    elsif(kar[$k] eq "f"){$kode="Ff "} #28
elsif(kar[$k] eq "g"){$kode="Gg "} #29    elsif(kar[$k] eq "h"){$kode="Hh "} #30
elsif(kar[$k] eq "j"){$kode="Jj "} #31    elsif(kar[$k] eq "k"){$kode="Kk "} #32
elsif(kar[$k] eq "l"){$kode="Ll "} #33    elsif(kar[$k] eq "m"){$kode="Mm "} #34
elsif(kar[$k] eq "n"){$kode="Nn "} #35    elsif(kar[$k] eq "p"){$kode="Pp "} #36
elsif(kar[$k] eq "q"){$kode="Qq "} #37    elsif(kar[$k] eq "r"){$kode="Rr "} #38
elsif(kar[$k] eq "s"){$kode="Ss "} #39    elsif(kar[$k] eq "t"){$kode="Tt "} #40
elsif(kar[$k] eq "v"){$kode="Vv "} #41    elsif(kar[$k] eq "w"){$kode="Ww "} #42
elsif(kar[$k] eq "x"){$kode="Xx "} #43    elsif(kar[$k] eq "z"){$kode="Zz "} #44
elsif(kar[$k] eq "="){$kode="Kt "} #45    elsif(kar[$k] eq "´"){$kode="Af "} #46
```

Bibliografie

[ABP08a] E. Alfonseca, S. Bilac, en S. Pharies. Decompounding query keywords from compounding languages. In *Proceedings of ACL-08: HLT, Short Papers (Companion Volume)*, pages 253–256. Association for Computational Linguistics, 2008. Beskikbaar by: `http://aclweb.org/anthology-new/P/P08/P08-1065.pdf`.

[ABP08b] E. Alfonseca, S. Bilac, en S. Pharies. German Decompounding in a Difficult Corpus. In *A. Gelbukh (Ed.): CICLing*, pages 128–139. Springer-Verlag Berlin, 2008. Beskikbaar by: `http://aclweb.org/anthology-new/P/P08/P08-1065.pdf`.

[ADAL00] M. Adda-Decker, G. Adda, en L. Lamel. Investigating text normalization and pronunciation variants for German broadcast transcription. In *Proceedings of ICSLP – 2000*, pages 266–269. Association for Computational Linguistics, 2000. Beskikbaar by: `ftp://tlp.limsi.fr/public/icslp00_h4ger.ps.Z`.

[Ant01] D. Antoš. PatLib, Pattern Manipulating Library. Master's thesis, Masaryk UniversityBrno, 2001.

[AS] D. Antoš en P. Sojka. Pattern Generation Revisited. In *Proceedings of the 16th European TEX Conference*.

[Bar02] W. Barth. Ein schnes Schrift bild erzeugen mit der Sicheren Sinnentsprechenden Silbentrennung SiSiSi, 2002. Beskikbaar by: `https://www.ads.tuwien.ac.at/research/SiSiSi/Si3Anleitung.hyph.pdf`.

[BFOS84] L. Breiman, J. Friedman, R. Ohlsen, en C. Stone. *Classification and Regression Trees*. Wadsworth International Group, 1984.

[BKC08] S. Bartlett, G. Kondrak, en C. Cherry. Automatic Syllabification with Structured SVMs for Letter-To-Phoneme Conversion. In *ACL-08: HLT*, pages 568–576. Association for Computational Linguistics, 2008. Beskikbaar by: `http://aclweb.org/anthology-new/P/P08/P08-1065.pdf`.

[BKS00] Barth, Kodydek, en Schonhacker. Reliable and Sense-Conveying Hyphenation for the German Language (SiSiSi). 2000. Beskikbaar by: `http://www.apm.tuwein.ac.at/research/SiSiSi.htm`.

[Bro02] R. D. Brown. Corpus-driven splitting of compound words. In *Proceedings of the Ninth International Conference on Theoretical and Methodological Issues in Ma-*

chine Translation (TMI-2002), pages 616–624. Association for Computational Linguistics, 2002. Beskikbaar by: `http://aclweb.org/anthology-new/P/P08/P08-1065.pdf`.

[Buc13] Moulton William G. Buccini, Anthony F. Brittanica - academic edition, cited April 2013. Beskikbaar by: `http://www.britannica.com/EBchecked/topic/231026/Germanic-languages`.

[Dic13] Business Dictionary. Gini index, 2013. Beskikbaar by: `http://www.businessdictionary.com/definition/gini-index.html`.

[Die09] Die Taalkommissie van die Suid-Afrikaanse Akademie vir Wetenskap en Kuns. *Afrikaanse woordelys en spelreëls*. Pharos Woordeboeke, NB-Uitgewers Beperk, Kaapstad, 2009.

[DvdB92] Walter Daelemans en Antal van den Bosch. Generalization performance of back-propagation learning on a syllabification task. In *Proceedings of the 3rd Twente Workshop on Language Technology*, pages 27–37, 1992.

[FF10] F. Fritzinger en A. Fraser. How to Avoid Burning Ducks: Combining Linguistic Analysis and Corpus Statistics for German Compound Processing. In *Proceedings of the Joint 5th Workshop on Statistical Machine Translation and MetricsMATR*, pages 224–234. Association for Computational Linguistics, 2010.

[Fic02] M. Fick. Neurale Netwerke as moontlike Woordafkappingstegniek vir Afrikaans. Master's thesis, Universiteit van Suid-Afrika, 2002.

[Fic03] M. Fick. Neurale Netwerke as moontlike Woordafkappingstegniek vir Afrikaans. *Die Suid-Afrikaanse Tydskrif vir Natuurwetenskap en Tegnologie*, 22(1), 2003.

[Fra98] Neil Fraser. Introduction to Neural Networks, 1998. Beskikbaar by: `http://vv.carleton.ca/~neil/neural/neuron-a.html`.

[GSM01] Dragan Gamberger, Tomislav Smuc, en Ivan Maric. DMS Tutorial - Decision Trees. 2001. Beskikbaar by: `http://dms.irb.hr/tutorial/tut_dtrees.php`.

[Hag96] Demuth H. B. Beale M. Hagan, M. T. *Neural Network Design*. PWS Publishing Company, 1996.

[Hay09] S. Haykin. *Neural Networks and Learning Machines*. Pearson Education, Inc., uitgawe 3, 2009.

[HGFO08] H. Hamilton, E. Gurak, L. Findlater, en W. Olive. Overview of Decision Trees. 2008. Beskikbaar by: `http://www2.cs.uregina.ca/~dbd/cs831/notes/ml/dtrees/4_dtrees1.html`.

[Ima13] Google Images, 2013. Beskikbaar by: `https://www.google.com/`.

[KAH08] P. Koehn, A. Arun, en H. Hoang. Towards better Machine Translation Quality for German–English Language Pairs. In *Proceedings of the Third Workshop on Statisti-*

cal Machine Translation, pages 139–142. Association for Computational Linguistics, 2008. Beskikbaar by: `http://aclweb.org/anthology-new/P/P08/P08-1065.pdf`.

[KK03] P. Koehn en K. Knight. Empirical methods for compound splitting. In *Proceedings of the tenth conference of EACL*, pages 187–193. Association for Computational Linguistics, 2003. Beskikbaar by: `http://aclweb.org/anthology-new/P/P08/P08-1065.pdf`.

[Knu84] Donald E. Knuth. *The TeXbook*. Addison-Wesley, 1984.

[Kod00] Gabriele Kodydek. A Word Analysis System for German Hyphenation, Full Text Search, and Spell Checking, with Regard to the Latest Reform of German Orthography. In *Proceedings of the Third International Workshop on Text, Speech and Dialogue—TSD 2000*, pages 39–44. Springer-Verlag, 2000.

[Kon98] Igor Kononenko. The Minimum Description Length Based Decision Tree Pruning. 1998. Beskikbaar by: `lkm.fri.uni-lj.si/xaigir/slo/clanki/pricai98.ps`.

[KS95] Igor Kononenko en Edvard Simec. Induction of decision trees using RELIEFF. *Citeseer*, 1995. Beskikbaar by: `http://citeseer.ist.psu.edu/cache/papers/cs/1252/http:zSzzSzai.fri.uni-lj.sizSzpaperszSzkononenko94-issek.pdf/kononenko95induction.pdf`.

[KS03] G. Kodydek en M Schönhacker. Si3trenn and si3silb: Using the sisisi word analysis system for pre-hyphenation and syllable counting in german documents. In *TSD, 2003*, pages 66–73, 2003. Beskikbaar by: `https://www.ads.tuwien.ac.at/publications/bib/pdf/kodydek-03.pdf`.

[LB91] F. M. Liang en P. Breitenlohner. PATtern GENeration program for the TEX 82 hyphenator. *Electronic documentation of PATGEN program version 2.0 from UNIXTEX distribution*, 1991. Beskikbaar by: `ftp://ftp.cs.umb.edu`.

[LB96] F. M. Liang en P. Breitenlohner. PATGEN.WEB in Text Format. 1996. Beskikbaar by: `http://ftp.cs.stanford.edu/pub/tex/unsupported/textware/patgen.web`.

[LG12] L. Le Grange. Veranderinge in skoolbiologie in Suid-Afrika ná apartheid. *Suid-Afrikaanse Tydskrif vir Natuurwetenskap en Tegnologie*, Volume 31(1):78 – 85, 2012. Beskikbaar by: `http://www.satnt.ac.za/index.php/satnt/article/view/385/html`.

[Lia83] F. M. Liang. *Word Hy-phen-a-tion by Com-pu-ter*. PhD thesis, Stanford University, 1983.

[MAD09] Y. Marchand, C. R. Adsett, en R. I. Damper. Automatic Syllabification in English: A Comparison of Different Algorithms. *Language and Speech*, 52(1):1–27, 2009. Beskikbaar by: `http://las.sagepub.com/cgi/content/refs/52/1/1` [laaste toegang 4 November 2009].

[Mat13] MathWorks. Neural network toolbox: Create, train, and simulate neural networks, 2013. Beskikbaar by: `http://www.mathworks.com/products/neural-network`.

[MDR02] C. Monz en M. De Rijke. Shallow Morphological Analysis in Monolingual Information Retrieval for Dutch, German, and Italian. In *C. A. Peters et al (Ed.): CLEF 2001, LNCS*, volume 2406, pages 262–277. Springer-Verlag Berlin, 2002.

[Mie13] Mieliestronk, 2013. Beskikbaar by: `http://mieliestronk.com/bruinmense.html`.

[MRS08] C. D. Manning, P. Raghavan, en H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008. Beskikbaar by: `http://nlp.stanford.edu/IR-book/information-retrieval-book.html`.

[Nil96] Nils J. Nilsson. Introduction to Machine Learning. 1996. Beskikbaar by: `http://robotics.stanford.edu/people/nilsson/MLDraftBook/ch6-ml.pdf`.

[NP13] CTexT (NWU-Pukke). Afrikaanse skryfgoed 4, 2013. Beskikbaar by: `http://speltoetser.co.za/index.php?frontend_action=display_text_content&item_id=131`.

[OG09] F. F. Odendaal en R. H. Gouws. *HAT Handwoordeboek van die Afrikaanse Taal*. Pearson Education South Africa, 2009.

[Paz13] Jan Pazdziora. Tex-hyphen, 2013. Beskikbaar by: `http://search.cpan.org/dist/TeX-Hyphen/`.

[Pha05] Pharos. *Afrikaans-Engels Woordeboek/English-Afrikaans Dictionary*. Pharos Woordeboeke, NB-Uitgewers Beperk, 2005.

[Pha13] Pharos. Speltoetser en woordafbreker, 2013. Beskikbaar by: `http://www.pharos.co.za/Books/10901`.

[PPvH08] S. Pilon, M.J. Puttkammer, en G.B. van Huyssteen. The development of a hyphenator and compound analyser for Afrikaans/Die ontwikkeling van 'n woordafbreker en kompositumanaliseerder vir Afrikaans. *Literator: Journal of Literacy Criticism, comparative linguistics and literary studies*, 2008.

[PSN06] M. Popović, D. Stein, en H. Ney. Statistical machine translation of German compound words. In *Proceedings of FinTAL – 5th International Conference on Natural Language Processing*, pages 616–624. Association for Computational Linguistics, 2006. Beskikbaar by: `http://aclweb.org/anthology-new/P/P08/P08-1065.pdf`.

[Qui86] J. R. Quinlan. Induction of Decision Trees. *Machine Learning*, 1986.

[SA] Petr Sojka en David Antoš. Context Sensitive Pattern Based Segmentation: A Thai Challenge. In *Proceedings of EACL 2003 workshop Computational Linguistics for South Asian Languages – Expanding Synergies with Europe*.

[SAH13] SAHO. South african history online: towards a people's history, 2013. Beskikbaar by: http://www.sahistory.org.za/dated-event/afrikaans-becomes-official-language-union-south-africa.

[Sch05] A. Schiller. German Compound Analysis with wfsc. In *Finite State Methods and Natural Language Processing*, pages 128–139. Springer, 2005.

[SS96] P. Smrž en P. Sojka. Word hy-phen-a-tion by Neural Networks. *FI MU Report Series*, August 1996.

[Stu13] Studentpolyglot.wordpress.com. Just a little something about afrikaans, 2013. Beskikbaar by: http://studentpolyglot.wordpress.com/2013/05/04/just-a-little-something-about-afrikaans/.

[Sys13] Salford Systems. CART Classification and Regression Trees, 2013. Beskikbaar by: http://www.salford-systems.com/products/cart.

[Wik13a] Wikipedia, 2013. Beskikbaar by: http://en.wikipedia.org/wiki/Computational_linguistics.

[Wik13b] Wikipedia. Afrikaans, 2013. Beskikbaar by: http://af.wikipedia.org/wiki/Afrikaans.

[Wik13c] Wikipedia. Afrikaanse woordelys en spelreëls, 2013. Beskikbaar by: http://af.wikipedia.org/wiki/Afrikaanse_Woordelys_en_Spelre\"els.

[Wik13d] Wikipedia. Decision Trees, 2013. Beskikbaar by: http://en.wikipedia.org/wiki/Decision_tree.

[Wik13e] Wikipedia. Germanic languages, 2013. Beskikbaar by: http://indoeuro.bizland.com/tree/germ/ger.html.

[Wik13f] Wikipedia. ID3 algorithm, 2013. Beskikbaar by: http://en.wikipedia.org/wiki/ID3_algorithm.

[Wik13g] Wikipedia. Machine learning, 2013. Beskikbaar by: http://http://en.wikipedia.org/wiki/Machine_learning.